Polynomials Regression

Lecture 14

Alexandra Chronopoulou

ILLINOIS

COLLEGE OF LIBERAL ARTS & SCIENCES

Department of Statistics
101 Illini Hall, MC-374
725 S. Wright St.
Champaign, IL 61820-5710

Learning objectives

In this lecture we will:

- discuss about polynomial regression
- introduce cubic splines
- perform splines rergession

General Additive Models

- The multiple linear model we have seen so far is formulated as

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \varepsilon_i,$$

with $i = 1, \ldots, n$, and ε_i satisfying the usual assumptions.

- A more general linear additive model can be written as

$$y_i = X_i^* \beta + f_1(x_{1i}) + f_2(x_{2i}) + f_3(x_{3i}, x_{4i}) + \ldots + \varepsilon_i$$

where X_i^* is a row of the design matrix that contains the parametric model components, β is the corresponding parameter vector and f_j are *smooth functions* of the x_k's.

Univariate Smooth Function

Consider a model containing only one predictor of the form

$$y_i = f(x_i) + \varepsilon_i$$

where

- y_i is a response variable ,
- x_i is a predictor,
- f is a *smooth* function, and
- ε_i are IID $\mathcal{N}(0, \sigma^2)$ random variables.

Polynomial Basis

Basis Functions

$$y_i = f(x_i) + \varepsilon_i$$

- Our goal is to estimate f using methods that we have already discussed.
- This implies that we need to represent f in such a way that the model above becomes a *linear* model.
- This can be done by choosing a *basis*, defining the space of functions of which f is an element.
- Once we choose the basis functions, they will be treated as **completely known**.

If $b_j(x)$ is the jth such basis function, then f is assumed to have the following representation

$$f(x) = \sum_{i=1}^{d} b_j(x)\beta_j$$

for some values of the unknown parameter β_j. Therefore, we managed to write our initial model as a **linear model**

$$y_i = \beta_0 + \sum_{j=1}^{d} b_j(x_i)\beta_j + \varepsilon_i$$

A Polynomial Basis

Suppose that f is believed to be a 4th order polynomial, so the space of polynomials of order 4 and below contains f.

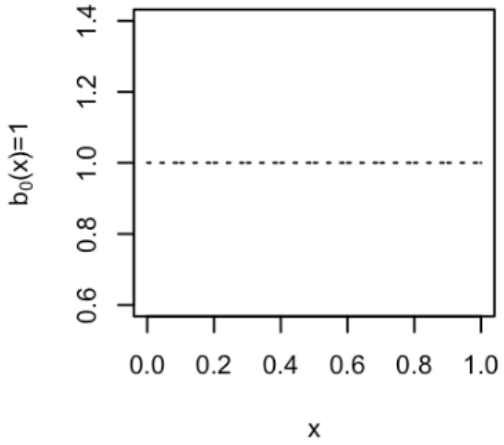
A basis for this space is

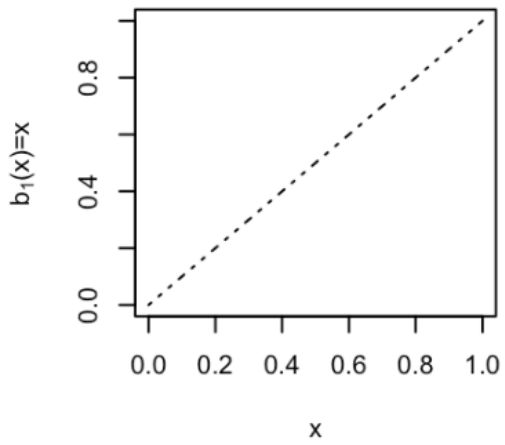
$$
\begin{aligned}
b_0(x) &= 1 \\
b_1(x) &= x \\
b_2(x) &= x^2 \\
b_3(x) &= x^3 \\
b_4(x) &= x^4
\end{aligned}
$$

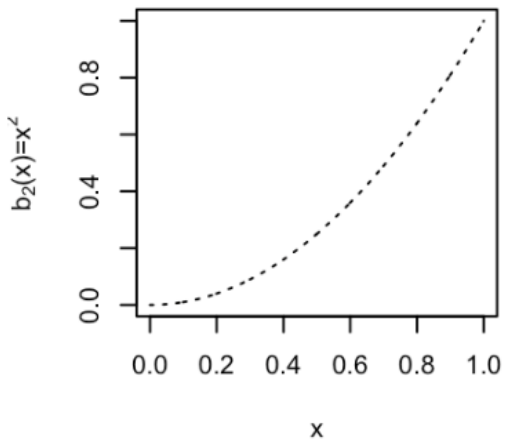
so that the model becomes

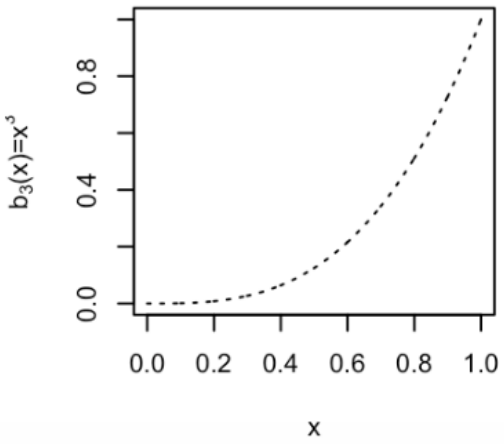
$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \varepsilon_i$$

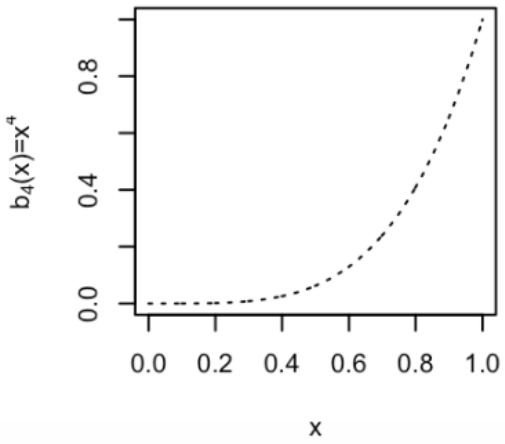
Illustration of Polynomial Basis Functions

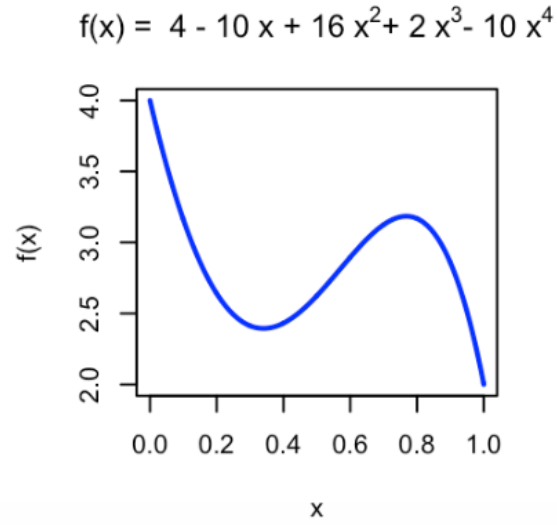












Polynomials Regression

Polynomials Regression

$$y_i = f(x_i) + \varepsilon_i \quad \longrightarrow \quad y_i = \beta_0 + \sum_{j=1}^{d} b_j(x_i)\beta_j + \varepsilon_i$$

where d is the degree of the polynomial component.

How do we choose d?

- Forward approach: Keep adding terms until the last added term is not significant.
- Backward approach: Start with a large d, and keep eliminating the terms that are not statistically significant, starting with the highest order term.

Order d of the Polynomial

Once we pick a value of d, then *should we test whether the other terms, x^j's with $j = 1, \ldots, d-1$, are significant or not*?

- Usually we **do not** test the significance of the lower-order terms.
- When we decide to use a polynomial of degree d, by default, *we include all the lower-order terms in our model*.

Order d of the Polynomial

Reasoning

In regression analysis, we do not want our results to be affected by a change of location/scale of the data. Consider the following *example*:

- Suppose the data $\{y_i, x_i\}_{i=1}^n$ are generated by the model:

$$y_i = x_i^2 + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

  But, are recorded as $\{z_i, x_i\}_{i=1}^n$, where $z_i = x_i + 2$, that is,

$$y_i = (z_i - 2)^2 + \varepsilon_i = 4 - 4z_i + z_i^2 + \varepsilon_i$$

- The linear term could become significant if we shift the x values.

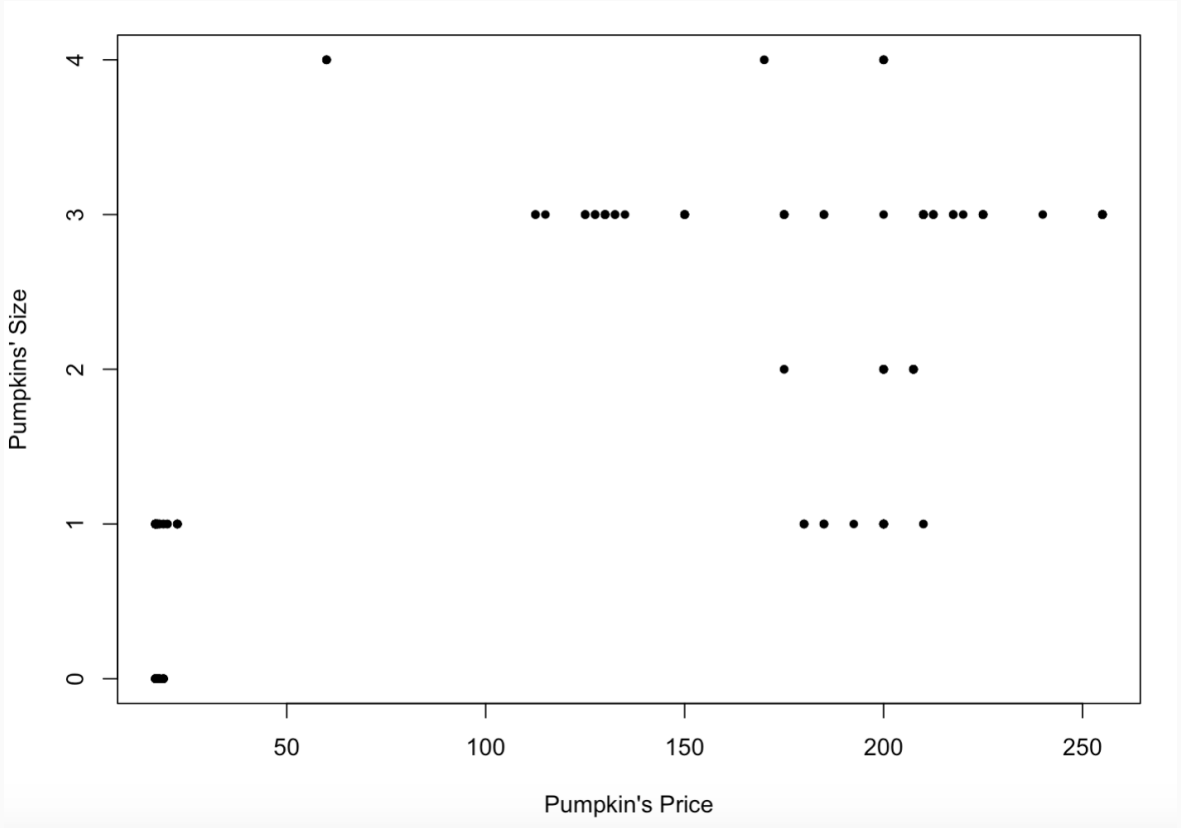
Exception

When we have a particular polynomial function in mind, e.g. a physics law.

Example: Chicago Pumpkins data set

The `pumpkins.csv` data set contains information regarding the *size* and *price* of pumpkins sold in the Chicago area. Our goal in this example is to *predict* the size of the pumpkin (response) based on its price (predictor).

The scatter plot of the data is shown below:



Example: Chicago Pumpkins data set

Forward Model Selection: $d = 4$

```
model3 =  lm(size ~ price + I(price^2) + I(price^3) + I(price^4), data=pumpkins)
summary(model3)
```

```
##
## Call:
## lm(formula = size ~ price + I(price^2) + I(price^3) + I(price^4),
##     data = pumpkins)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3200 -0.4497 -0.1241  0.5539  1.7925
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.871e+00  3.782e-01  -7.590 6.83e-13 ***
## price        2.314e-01  2.630e-02   8.800 2.60e-16 ***
## I(price^2)  -2.565e-03  3.785e-04  -6.776 9.25e-11 ***
## I(price^3)   1.061e-05  1.973e-06   5.378 1.76e-07 ***
## I(price^4)  -1.470e-08  3.443e-09  -4.271 2.80e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6941 on 243 degrees of freedom
## Multiple R-squared:  0.7073, Adjusted R-squared:  0.7025
## F-statistic: 146.8 on 4 and 243 DF,  p-value: < 2.2e-16
```

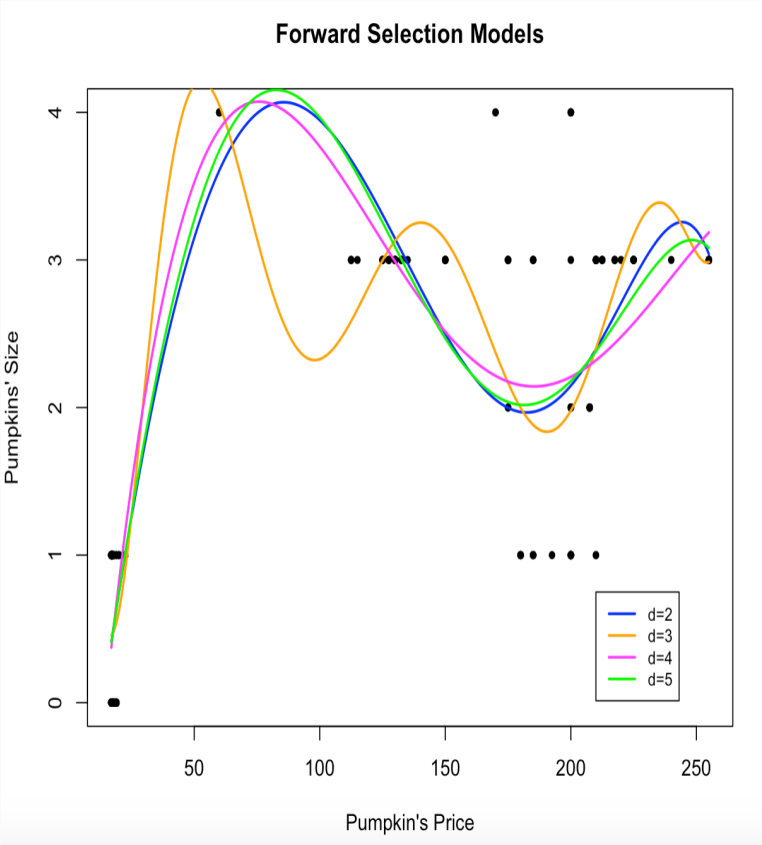
Example: Chicago Pumpkins data set

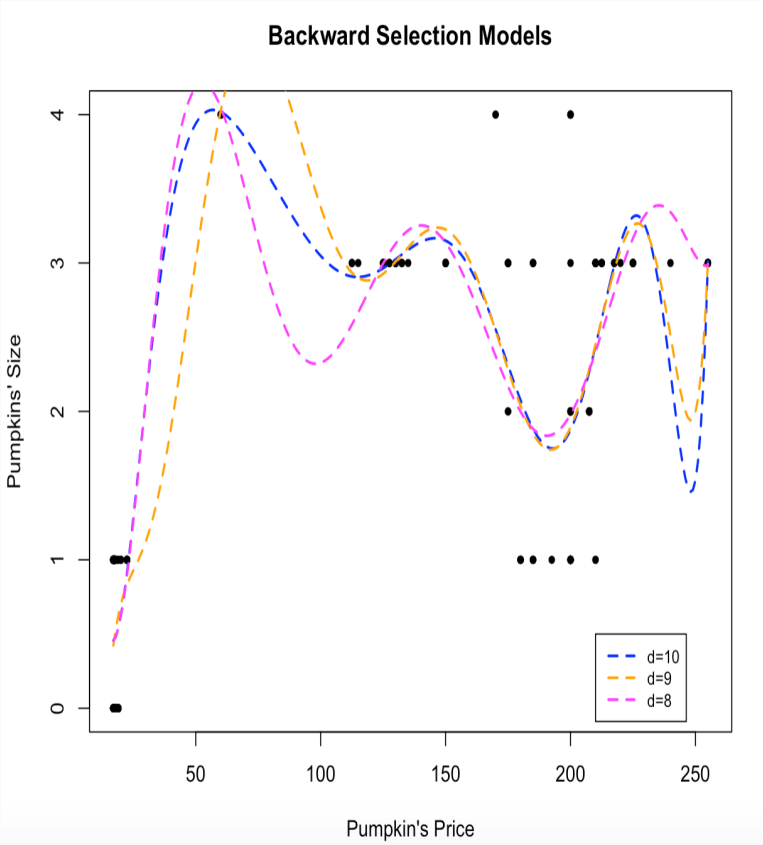
Backward Model Selection: $d = 8$

```
model_3 =  lm(size ~ price + I(price^2) + I(price^3)+ I(price^4)+ I(price^5)+ I(price^6) + I(price^7)
+ I(price^8), data=pumpkins)
summary(model_3)
```

```
##
## Call:
## lm(formula = size ~ price + I(price^2) + I(price^3) + I(price^4) +
##     I(price^5) + I(price^6) + I(price^7) + I(price^8), data = pumpkins)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.39988 -0.45678 -0.05827  0.53309  2.02119
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.140e+00  3.347e+00   2.731 0.006791 **
## price       -1.368e+00  4.257e-01  -3.215 0.001486 **
## I(price^2)   7.522e-02  2.032e-02   3.702 0.000266 ***
## I(price^3)  -1.798e-03  4.783e-04  -3.759 0.000214 ***
## I(price^4)   2.262e-05  6.154e-06   3.675 0.000293 ***
## I(price^5)  -1.611e-07  4.541e-08  -3.549 0.000466 ***
## I(price^6)   6.535e-10  1.918e-10   3.407 0.000771 ***
## I(price^7)  -1.406e-12  4.316e-13  -3.259 0.001282 **
## I(price^8)   1.246e-15  4.008e-16   3.108 0.002112 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6544 on 239 degrees of freedom
## Multiple R-squared:  0.7441, Adjusted R-squared:  0.7355
## F-statistic: 86.87 on 8 and 239 DF,  p-value: < 2.2e-16
```

Regression Lines for Chicago Pumpkins





Orthogonal Polynomials

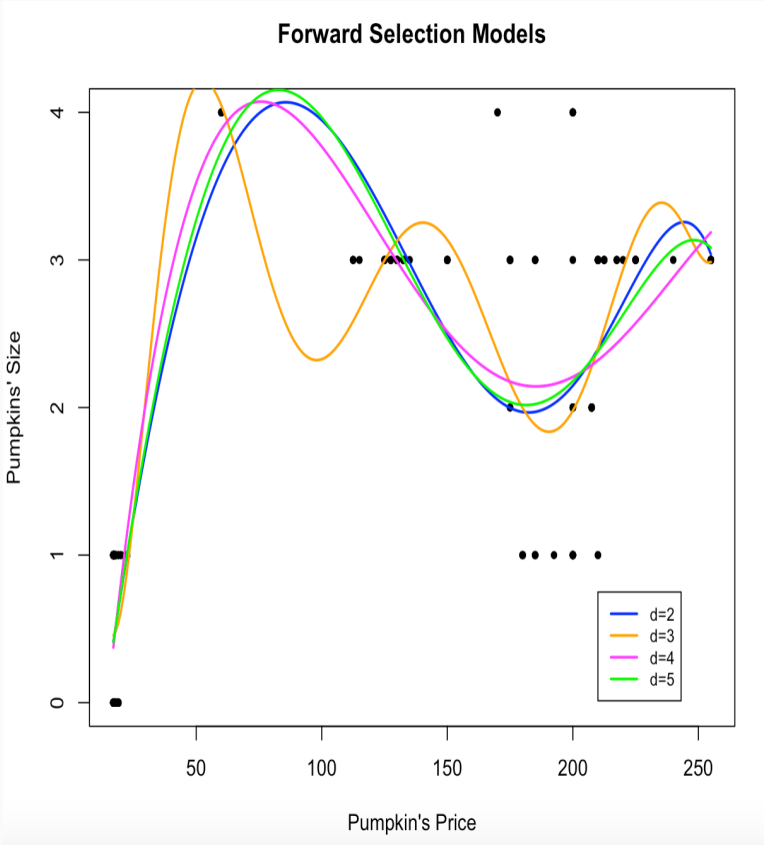
- Fitting high order polynomials is generally not recommended, since they are very unstable and difficult to interpret.
- Successive predictors x^j are highly correlated introducing multicollinearity problems.
- One way around this is to fit orthogonal polynomials of the form:

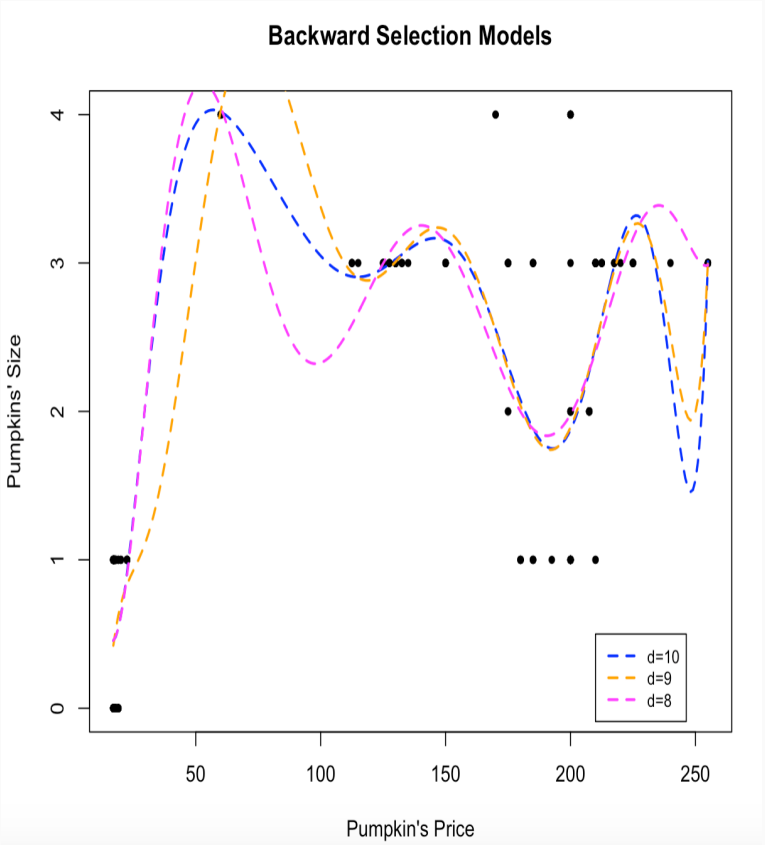
$$y_i = \beta_0 + \beta_1 z_1 + \ldots + \beta_d z_d + \varepsilon_i$$

where each $z_j = a_1 + b_2 x + \ldots + \kappa_j x^j$ is a polynomial of order j with coefficients chosen such that $z_i^\top z_j = 0$.

In `R`, use the function **poly(.)** to fit orthogonal polynomials.

Regression Lines for Chicago Pumpkins: Standard Polynomials





Piece-wise Polynomials

- If the true mean of $\mathbb{E}(Y|X = x) = f(x)$ is too wiggly, we might need to fit a higher order polynomial, which is not always a good idea.
- Instead we will consider piece-wise polynomials:
  - we divide the range of x into several intervals, and
  - within each interval $f(x)$ is a low-order polynomial, e.g., cubic or quadratic, but the polynomial coefficients change from interval to interval;
  - in addition we require the overall $f(x)$ to be continuous up to certain derivatives.

Cubic Splines

Splines

- Polynomials are smooth, but each point affects the fit globally.
- Piece-wise polynomials regression localizes the influence of each data point to its segment, but they are not smooth enough.
- Combination of beneficial aspects of both approaches: Splines!

Cubic Splines

- A Cubic Spline is a curve constructed from sections of cubic polynomials, joined together so that *the curve is continuous up to second derivative*.
- The points at which the sections join are called the **knots** of the spline.
- Typically, the knots would either be evenly spaced through the range of observed x values, or placed at the quantiles of the distribution of unique x values.
- Each section of cubic has different coefficients, but at the knots *it will match its neighboring sections in value and first two derivatives*.

Cubic Splines Definition

We want to define a cubic spline function in the interval $[a, b]$

- Define m knots such that: $a < \xi_1 < \xi_2 < \ldots < \xi_m < b$
- A function g defined on $[a, b]$ is a cubic spline with respect to knots $\{\xi_i\}_{i=1}^{m}$ if:
  1. g is a cubic polynomial in each of the $m + 1$ intervals,

     $$g(x) = d_i x^3 + c_i x^2 + b_i x + a_i, \quad x \in [\xi_i, \xi_{i+1}]$$

     where $i = 0, \ldots, m$, $\xi_0 = a$ and $\xi_{m+1} = b$
  2. g is continuous up to the 2nd derivative: since g is continuous up to the 2nd derivative for any point inside an interval, it suffices to check the following conditions:

     $$g^{(0,1,2)}(\xi_i^+) = g^{(0,1,2)}(\xi_i^-), \quad i = 1 : m$$

     This expression indicates that the function and the first and second order derivatives are continuous at the knots.

How many free parameters do we need to represent a cubic spline?

(i) 4 parameters (d_i, c_i, b_i, a_i) for each of the $(m+1)$ intervals.

(ii) 3 constraints at each of the m knots (continuity constraints).

The total number of *free* parameters (similar to the number of *degrees of freedom*) is:

$$4(m+1) - 3m = m + 4$$

Some properties of the cubic splines

Suppose the knots $\{\xi_i\}_{i=1}^m$ are given.

- If $g_1(x)$ and $g_2(x)$ are cubic splines, the linear combination

$$a_1 g_1(x) + a_2 g_2(x)$$

  is also a cubic spline, where a_1 and a_2 are known constants.

That is, for a set of given knots, the corresponding cubic splines form a *linear space (of functions) with* dim $(m + 4)$.

A Cubic Splines Basis

– A set of basis functions for cubic splines (w.r.t knots $\{\xi_i\}_{i=1}^m$) is given by:

$$
\begin{aligned}
h_0(x) &= 1 \\
h_1(x) &= x \\
h_2(x) &= x^2 \\
h_3(x) &= x^3; \\
h_{i+3}(x) &= (x - \xi_i)_+^3, \quad i = 1, 2, \ldots, m
\end{aligned}
$$

That is, any cubic spline can be uniquely expressed as:

$$
\beta_0 + \sum_{j=1}^{m+3} \beta_j h_j(x)
$$

Another Cubic Splines Basis

- Given knot locations, there are many alternative, but equivalent ways of writing down a basis for cubic splines.
- For example, *another* basis for cubic splines can be the following:

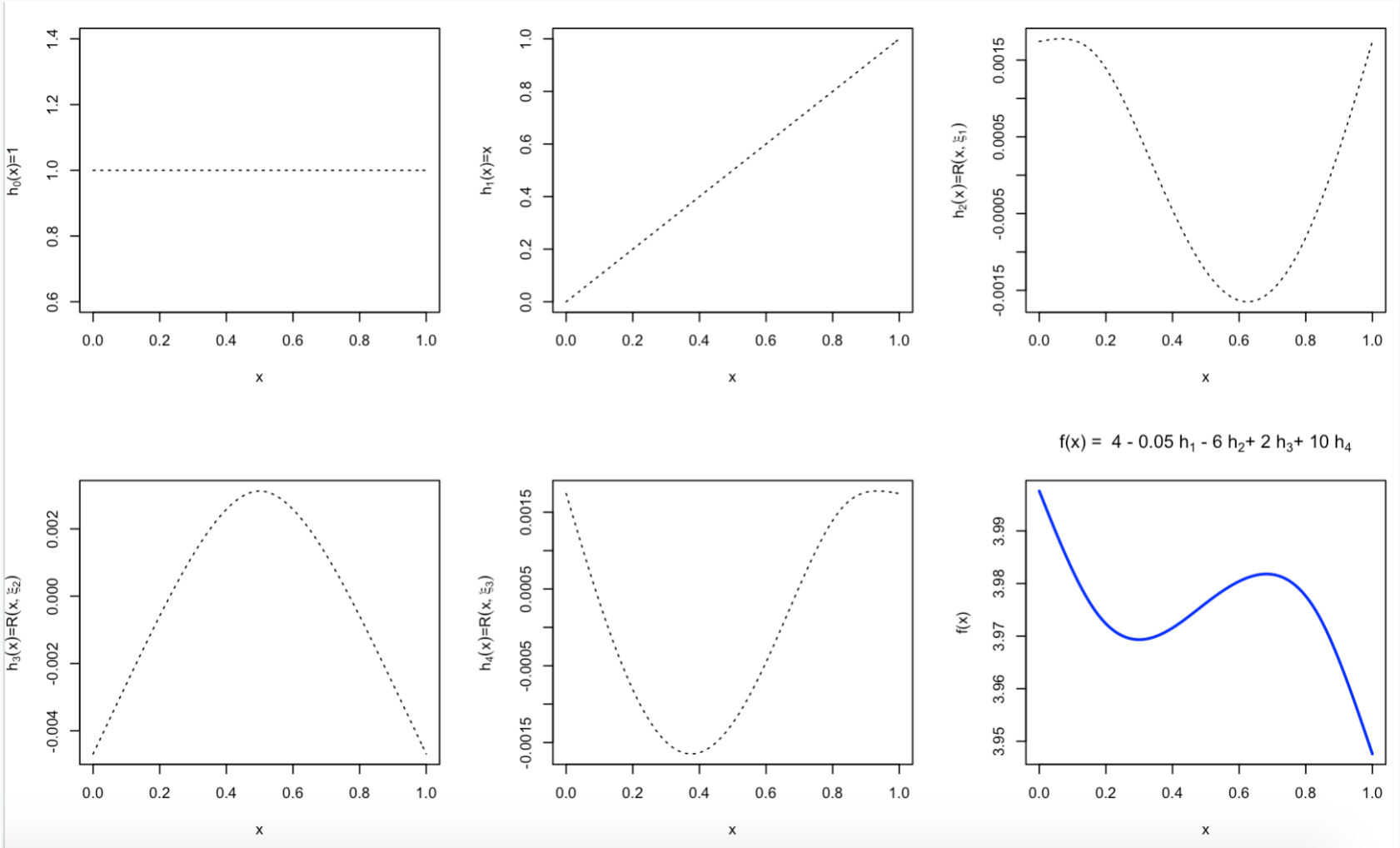
$$
\begin{aligned}
h_0(x) &= 1 \\
h_1(x) &= x \\
h_{i+1}(x) &= R(x, \xi_i^*), \ i = 1, \ldots, q-1
\end{aligned}
$$

where

$$
\begin{aligned}
R(x,z) = &\left[(z-1/2)^2 - 1/12\right]\left[(x-1/2)^2 - 1/12\right]/4 \\
&- \left[(|x-z|-1/2)^4 - 1/2(|x-z|-1/2)^2 + 7/240\right]/24
\end{aligned}
$$

Another Cubic Splines Basis: Illustration

$$h_0(x) = 1;\ h_1(x) = x;\ h_{i+1}(x) = R(x, \xi_i^*),\ i = 1, \ldots, 4$$



Blue line (last plot): $f(x) = 4h_0 - 0.05h_1 - 6h_2 + 2h_3 + 10h_4$

B-Splines Basis Functions in R

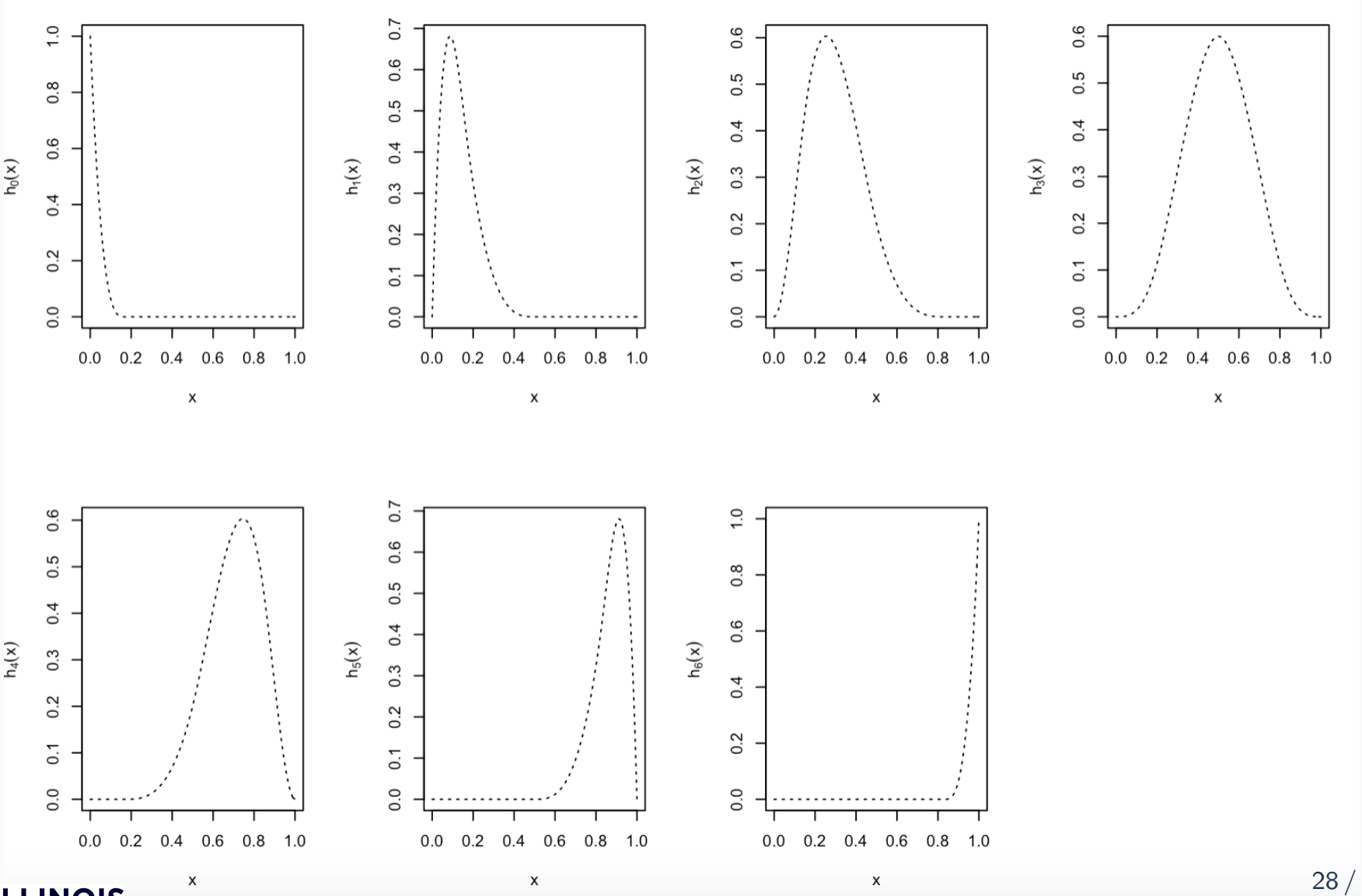
In **R**, we typically use the **B-Splines** basis. The R-function is `bs` and is part of the `splines` library.

```
bs(x, df, knots, degree=3, intercept=FALSE)
```

Understand how R counts the degrees-of-freedom

- The default `degree` is 3, which corresponds to a cubic polynomial.
- You can specify the location of `knots`, or
- You can specify the `df`.
  - Recall that a cubic spline with m knots has $m+4$ df, so this corresponds to $m = df - 4$ knots.
  - By default, **R** puts knots at the $1/(m+1), \ldots, m/(m+1)$ quantiles of $x_{1:n}$.
- The default `intercept` is FALSE, which is what we prefer when we input this object in the `lm` function.

B-Splines Illustration



Natural Cubic Splines (NCS)

- A cubic spline on $[a, b]$ is a Natural Cubic Spline if its *second and third derivatives are zero at a and b*.
- This condition implies that NCS is a linear function in the two extreme intervals $[a, \xi_1]$ and $[\xi_m, b]$. The linear functions in the two extreme intervals are completely determined by their neighboring intervals.
- The degree of freedom of NCS's with m knots is :

$$4(m+1) - 3m - 4 = m$$

(We have 4 additional constraints.)

Natural Cubic Splines Basis

A Natural Cubic Spline with m knots is represented by m basis functions, for example, one such basis is given by

$$N_1(x) = 1$$
$$N_2(x) = x$$
$$N_{k+2}(x) = d_k(x) - d_{k-1}(x)$$

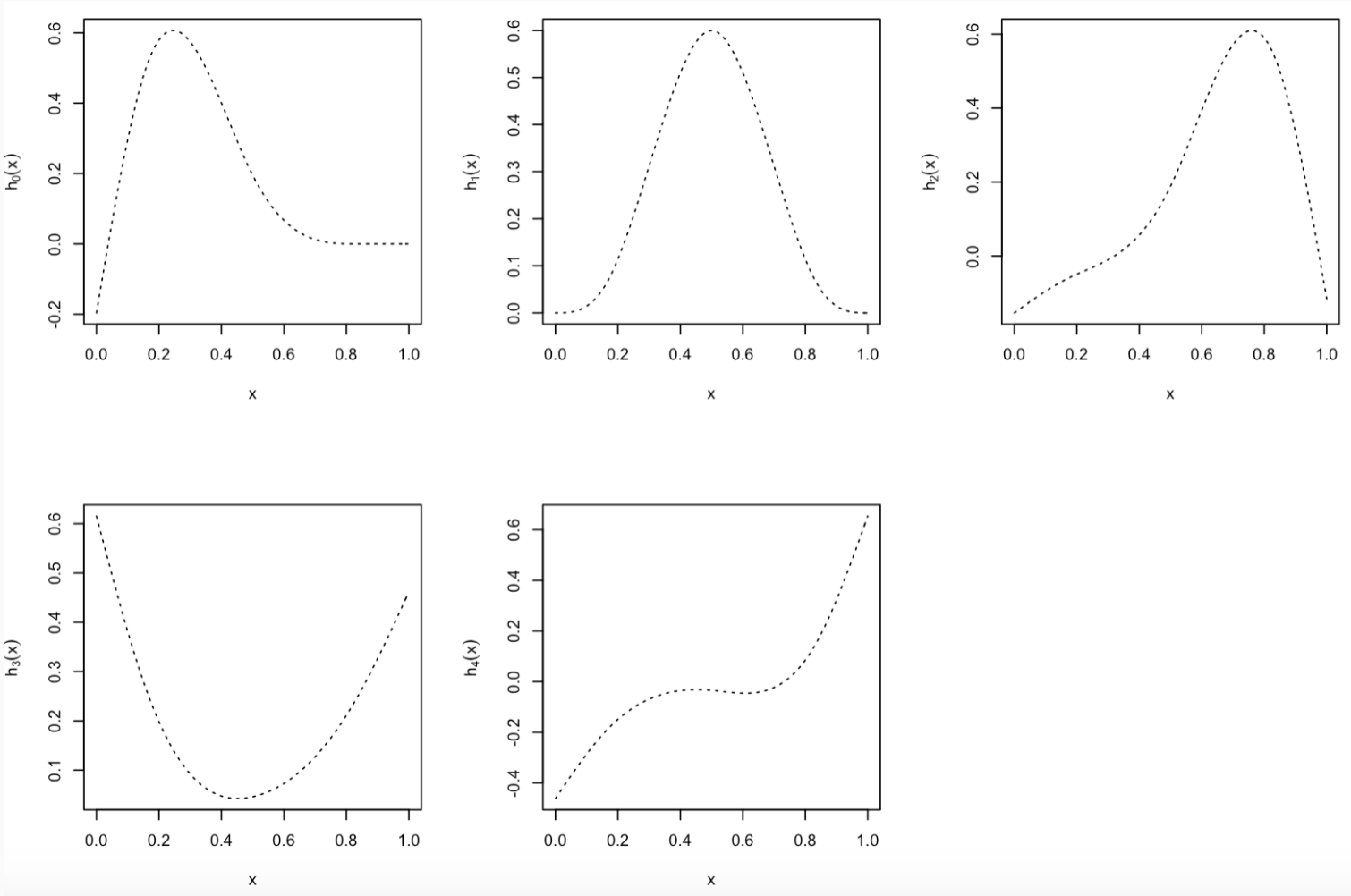
where

$$d_k(x) = \frac{(x - \xi_k)_+^3 - (x - \xi_m)_+^3}{\xi_m - \xi_k}$$

Each of these derivatives can be seen to have zero second and third derivative for $x \geq \xi_m$.

Natural Cubic Splines Illustration

```
ns(x, df, knots, intercept=TRUE, Boundary.knots)
```



NCS Basis functions

- To generate a NCS basis for a given set of x_i's, we use the command ns.
- Recall that the linear functions in the two extreme intervals are completely determined by the other cubic splines. So data points in the two extreme intervals (i.e., outside the two boundary knots) are wasted since they do not affect the fitting. Therefore, by default, **R** puts the two boundary knots as the min and max of the x_i's.
- You can tell **R** the location of knots, which are the *interior* knots. Recall that a NCS with m knots has m df. So, the df is equal to the number of (interior) knots plus 2, where 2 means the two boundary knots.
- Or you can tell **R** the df. If $intercept = TRUE$, then we need $m = df - 2$ knots, otherwise we need $m = df - 1$ knots. Again, by default, **R** puts knots at the $1/(m+1), ..., m/(m+1)$ quantiles of $x_{1:n}$.

Regression Splines

- We can represent the model on the observed n data points using matrix notation:

$$\underset{n\times 1}{\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}} = \underset{n\times p}{\begin{pmatrix} h_0(x_1) & h_1(x_1) & \dots & h_{p-1}(x_1) \\ h_0(x_2) & h_1(x_2) & \dots & h_{p-1}(x_2) \\ & & & \\ h_0(x_n) & h_1(x_n) & \dots & h_{p-1}(x_n) \end{pmatrix}} \underset{p\times 1}{\begin{pmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_p \end{pmatrix}}$$

  where our design matrix is the matrix **F** of basis functions.

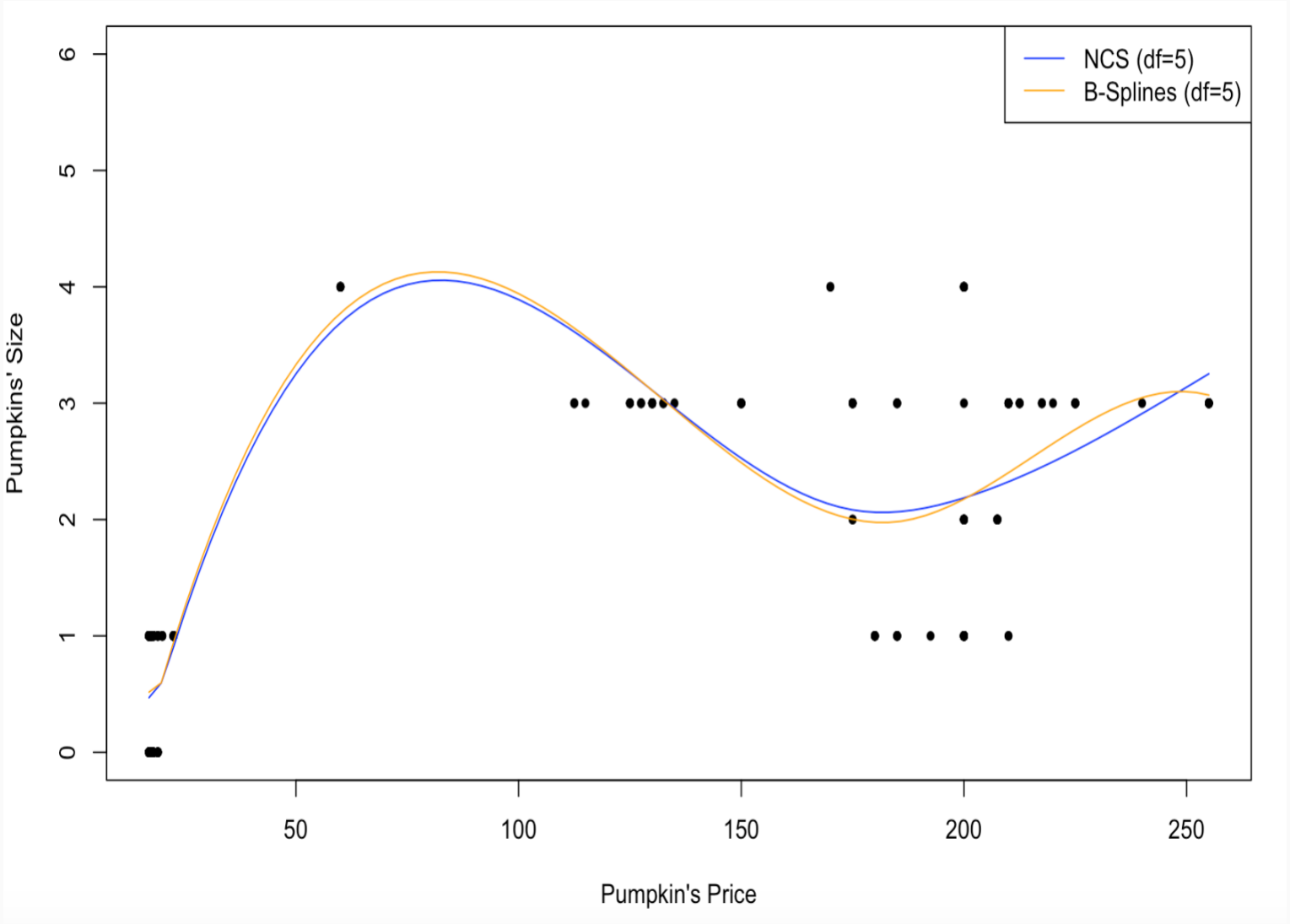
- We can find β by solving the problem:

$$\hat{\beta} = \arg\min_{\beta} ||\mathbf{y} - \mathbf{F}\beta||^2$$

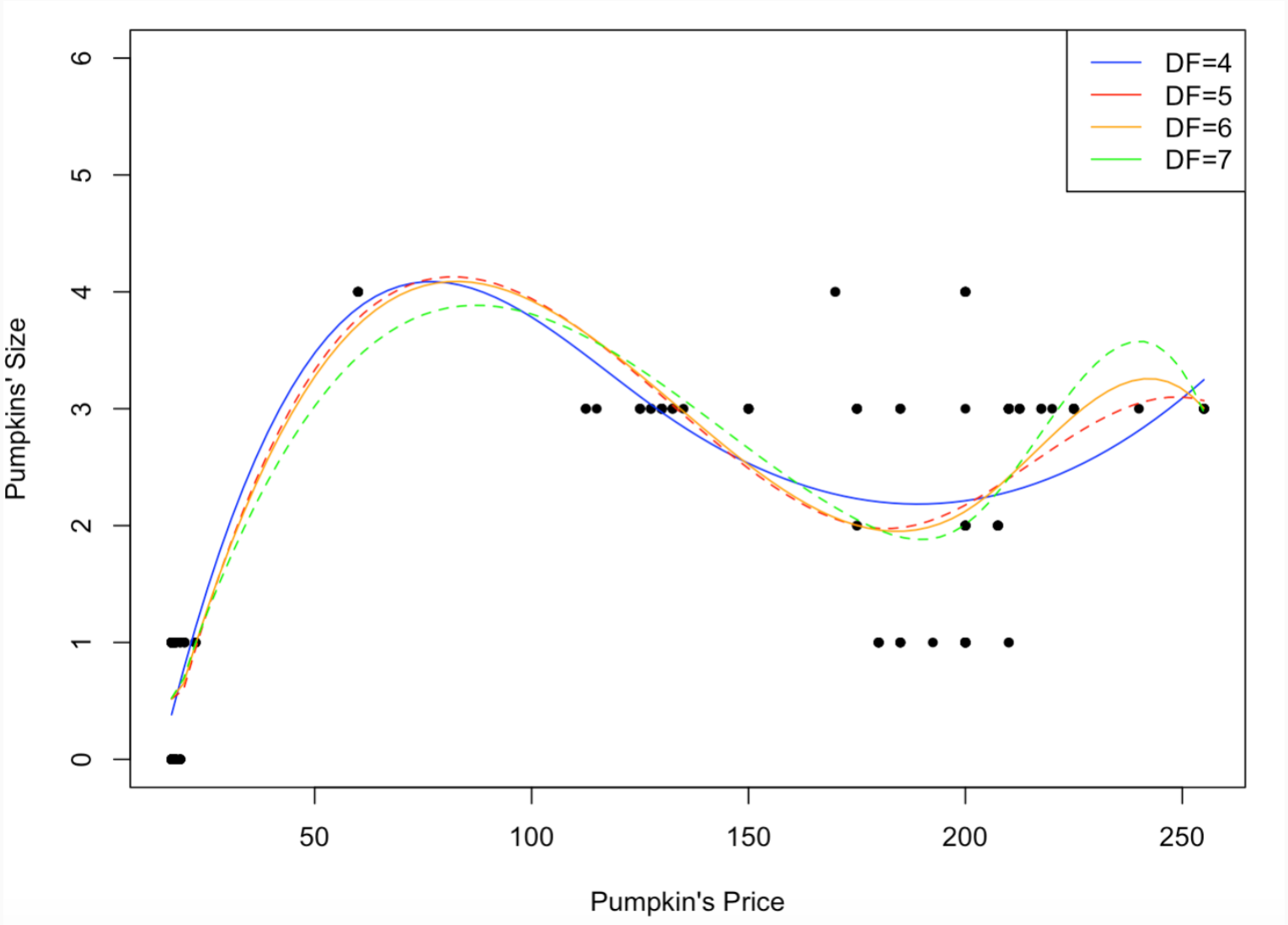
Regression Splines in R

- We can obtain the design matrix $\mathbf{F}$ by commands bs (B-splines) or ns (NCS) in **R**, and then call the regression function lm.
- To select the number of knots we can use K-fold cross-validation (CV) (More on this later).

Chicago Pumpkins data set (revisited)



Chicago Pumpkins data set (revisited)



How to select the optimal number of knots (or df)?

K-Fold Cross-Validation

1. Set a fixed number of knots (or df).
2. Divide the set of observations into k groups (or *folds*).
3. Leave the first fold as a validation set (not used to fit the model). Fit the Regression Spline with a fixed number of knots using the remaining $k - 1$ folds.
4. Calculate the Mean Square Error for fold 1: MSE_1.

K-Fold Cross-Validation

5. Repeat the previous steps k times. Each time a new validation set is used to calculate MSE_i.

6. Calculate the average k-fold Cross-Validation error: $CV(k) = \frac{1}{k} \sum_{i=1}^{k} MSE_i$.

7. Repeat 2 to 6 with a new number of knots (or df).

8. Select the number of knots that minimizes the k-fold CV error or $CV(k)$.

K-Fold Cross-Validation