Generalized Regression Splines in R

Georges Monette [1] John Fox [2]

[1]York University, Toronto, Canada

[2]McMaster University, Hamilton, Canada

useR! 2019, Toulouse, France

Introduction

- *Regression splines* are piecewise polynomials that are constrained to join smoothly at transition points called *knots*.

Introduction

- *Regression splines* are piecewise polynomials that are constrained to join smoothly at transition points called *knots*.
- Regression splines are traditionally introduced as an alternative to other methods for modeling nonlinear relationships (see, e.g., Fox, 2016; Harrell, 2015):

Introduction

- *Regression splines* are piecewise polynomials that are constrained to join smoothly at transition points called *knots*.
- Regression splines are traditionally introduced as an alternative to other methods for modeling nonlinear relationships (see, e.g., Fox, 2016; Harrell, 2015):
  - Unlike power transformations, regression splines can handle relationships that are nonmonotone or complex.

Introduction

- *Regression splines* are piecewise polynomials that are constrained to join smoothly at transition points called *knots*.
- Regression splines are traditionally introduced as an alternative to other methods for modeling nonlinear relationships (see, e.g., Fox, 2016; Harrell, 2015):
  - Unlike power transformations, regression splines can handle relationships that are nonmonotone or complex.
  - Unlike polynomial regression, regression splines are sensitive to local characteristics of the data.

Introduction

- *Regression splines* are piecewise polynomials that are constrained to join smoothly at transition points called *knots*.
- Regression splines are traditionally introduced as an alternative to other methods for modeling nonlinear relationships (see, e.g., Fox, 2016; Harrell, 2015):
  - Unlike power transformations, regression splines can handle relationships that are nonmonotone or complex.
  - Unlike polynomial regression, regression splines are sensitive to local characteristics of the data.
  - Unlike nonparametric regression, regression splines are fully parametric.

Introduction

- *Regression splines* are piecewise polynomials that are constrained to join smoothly at transition points called *knots*.
- Regression splines are traditionally introduced as an alternative to other methods for modeling nonlinear relationships (see, e.g., Fox, 2016; Harrell, 2015):
  - Unlike power transformations, regression splines can handle relationships that are nonmonotone or complex.
  - Unlike polynomial regression, regression splines are sensitive to local characteristics of the data.
  - Unlike nonparametric regression, regression splines are fully parametric.
- It is also usual *not* to focus on the estimated parameters for a regression spline and instead to represent the fit of the model graphically.

Introduction

- *Regression splines* are piecewise polynomials that are constrained to join smoothly at transition points called *knots*.
- Regression splines are traditionally introduced as an alternative to other methods for modeling nonlinear relationships (see, e.g., Fox, 2016; Harrell, 2015):
  - Unlike power transformations, regression splines can handle relationships that are nonmonotone or complex.
  - Unlike polynomial regression, regression splines are sensitive to local characteristics of the data.
  - Unlike nonparametric regression, regression splines are fully parametric.
- It is also usual *not* to focus on the estimated parameters for a regression spline and instead to represent the fit of the model graphically.
  - Traditional regression-spline bases, such as B-splines, are selected for numerical stability rather than for interpretability.

Introduction

- *Regression splines* are piecewise polynomials that are constrained to join smoothly at transition points called *knots*.
- Regression splines are traditionally introduced as an alternative to other methods for modeling nonlinear relationships (see, e.g., Fox, 2016; Harrell, 2015):
  - Unlike power transformations, regression splines can handle relationships that are nonmonotone or complex.
  - Unlike polynomial regression, regression splines are sensitive to local characteristics of the data.
  - Unlike nonparametric regression, regression splines are fully parametric.
- It is also usual *not* to focus on the estimated parameters for a regression spline and instead to represent the fit of the model graphically.
  - Traditional regression-spline bases, such as B-splines, are selected for numerical stability rather than for interpretability.
  - Although the emphasis on graphical interpretation makes sense, it also represents a missed opportunity.

Introduction

- We introduce *generalized regresson splines* and the `gspline()` function in the **carEx** package, which implements them.

Introduction

- We introduce *generalized regresson splines* and the `gspline()` function in the **carEx** package, which implements them.
- Our presentation consists of several parts:

Introduction

- We introduce *generalized regresson splines* and the `gspline()` function in the **carEx** package, which implements them.
- Our presentation consists of several parts:
  - A quick orientation to regression splines.

Introduction

- We introduce *generalized regresson splines* and the `gspline()` function in the **carEx** package, which implements them.
- Our presentation consists of several parts:
  - A quick orientation to regression splines.
  - A basic explanation of the implementation of regression splines in the `gspline()` function.

Introduction

- We introduce *generalized regresson splines* and the `gspline()` function in the **carEx** package, which implements them.
- Our presentation consists of several parts:
  - A quick orientation to regression splines.
  - A basic explanation of the implementation of regression splines in the `gspline()` function.
  - An explanation of the linear algebra underlying this implementation.

Introduction

- We introduce *generalized regresson splines* and the `gspline()` function in the **carEx** package, which implements them.
- Our presentation consists of several parts:
  - A quick orientation to regression splines.
  - A basic explanation of the implementation of regression splines in the `gspline()` function.
  - An explanation of the linear algebra underlying this implementation.
  - An illustrative application.

Introduction

- We introduce *generalized regresson splines* and the `gspline()` function in the **carEx** package, which implements them.
- Our presentation consists of several parts:
  - A quick orientation to regression splines.
  - A basic explanation of the implementation of regression splines in the `gspline()` function.
  - An explanation of the linear algebra underlying this implementation.
  - An illustrative application.
  - Further thoughts on implementing generalized regression splines in R.

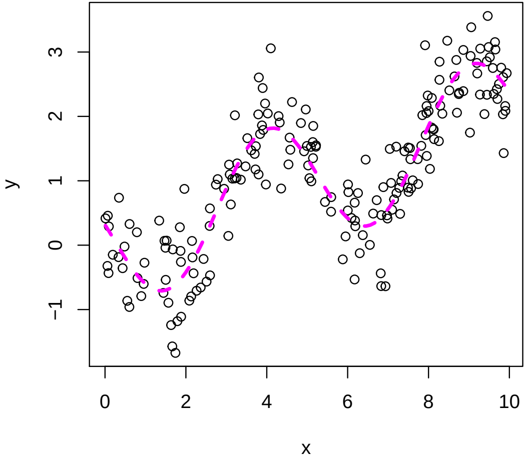
Introduction

- We introduce *generalized regresson splines* and the `gspline()` function in the **carEx** package, which implements them.
- Our presentation consists of several parts:
  - A quick orientation to regression splines.
  - A basic explanation of the implementation of regression splines in the `gspline()` function.
  - An explanation of the linear algebra underlying this implementation.
  - An illustrative application.
  - Further thoughts on implementing generalized regression splines in R.
- To install the **carEx** package:
  `install.packages("carEx", repos="http://R-Forge.R-project.org")`.

Regression Spline Basics

- To provide a point of reference, we randomly generate $n = 200$ observations according to the simple nonlinear regression model

$$x \sim \text{unif}(0, 10)$$
$$\varepsilon \sim N(0, 0.5^2)$$
$$y = \cos(1.25(x + 1)) + x/5 + \varepsilon$$

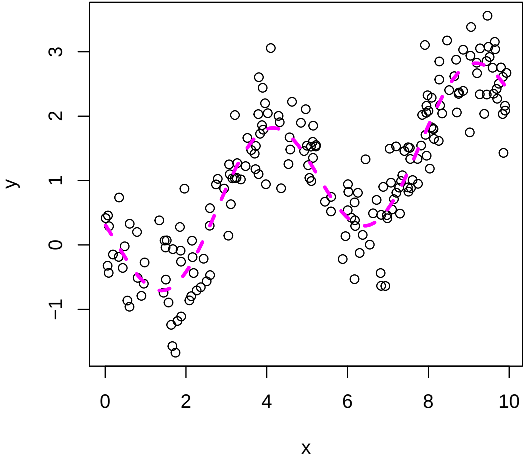


Regression Spline Basics

- To provide a point of reference, we randomly generate $n = 200$ observations according to the simple nonlinear regression model

$$x \sim \text{unif}(0, 10)$$
$$\varepsilon \sim N(0, 0.5^2)$$
$$y = \cos(1.25(x+1)) + x/5 + \varepsilon$$

- The broken line on the graph shows $E(y|x) = \cos(1.25(x+1)) + x/5$.

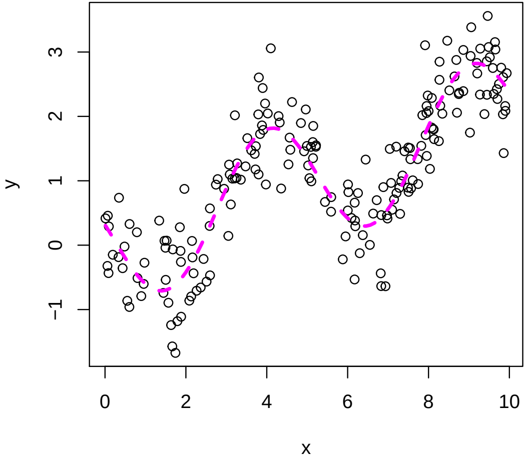


Regression Spline Basics

- To provide a point of reference, we randomly generate $n = 200$ observations according to the simple nonlinear regression model

$$x \sim \text{unif}(0, 10)$$
$$\varepsilon \sim N(0, 0.5^2)$$
$$y = \cos(1.25(x + 1)) + x/5 + \varepsilon$$

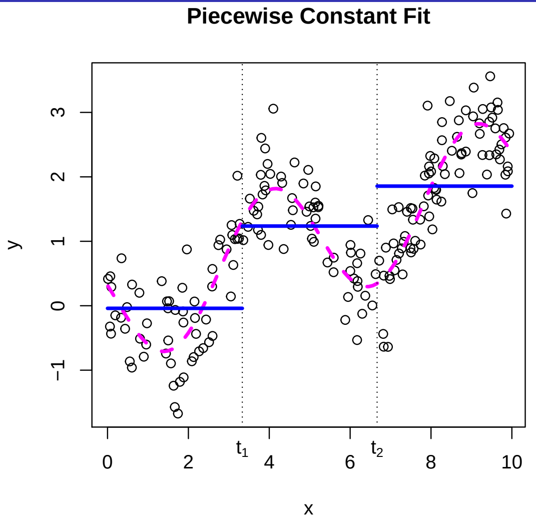
- The broken line on the graph shows $E(y|x) = \cos(1.25(x + 1)) + x/5$.
- We introduce a succession of increasingly complex segmented polynomial regression models, culminating in traditional cubic regression splines.



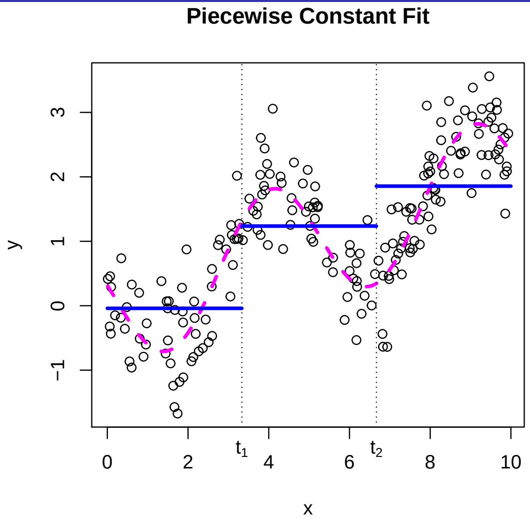
- *Piecewise polynomials* begin by dividing the range of x into non-overlapping intervals at k ordered x-values t_j called *knots*: $(-\infty, t_1], (t_1, t_2], \ldots, (t_k, \infty)$.

- *Piecewise polynomials* begin by dividing the range of x into non-overlapping intervals at k ordered x-values t_j called *knots*: $(-\infty, t_1], (t_1, t_2], \ldots, (t_k, \infty)$.
- Then a degree-p polynomial is fit by least-squares regression to the data in each interval.

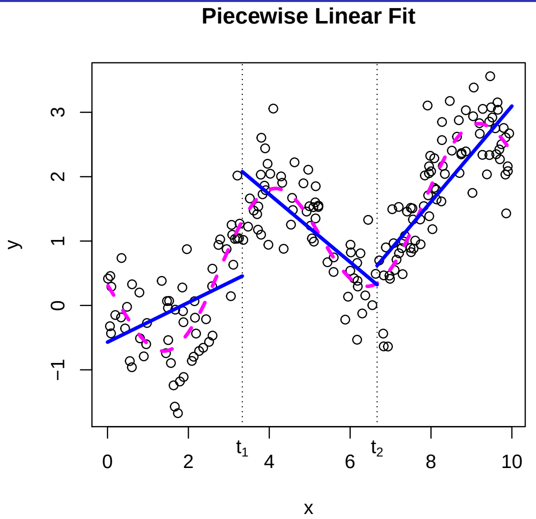
- The simplest case is a degree-0 polynomial, which generates a *piecewise-constant* fit by computing the mean *y*-value in each interval.



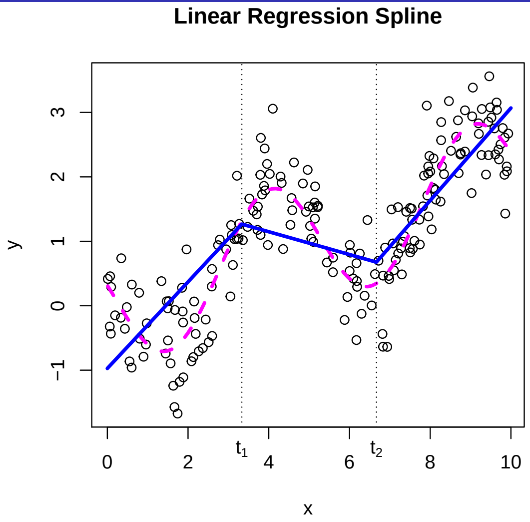
- The simplest case is a degree-0 polynomial, which generates a *piecewise-constant* fit by computing the mean y-value in each interval.
- For the example, we set $k = 2$ knots at $t_1 = 10/3$ and $t_2 = 2 \times 10/3$, which are the $1/3$ and $2/3$ quantiles of the uniform$[0, 10]$ distribution from which the x-values were generated.



- A next step is to generalize the model to a *piecewise-linear* fit (that is, a piecewise degree-1 polynomial).



- We convert the piecewise-linear fit into a *linear regression spline* by constraining the regression lines on the two sides of each knot to be equal at the knot.



- We can do this by fitting the linear model

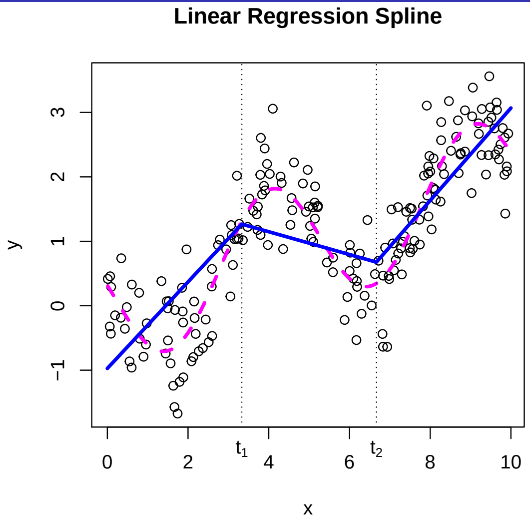
$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \varepsilon_i$$

where $x_{i1} = x_i$,

$$x_{i2} = \begin{cases} 0 & \text{for } x_i \leq t_1 \\ x_i - t_1 & \text{for } x_i > t_1 \end{cases}$$

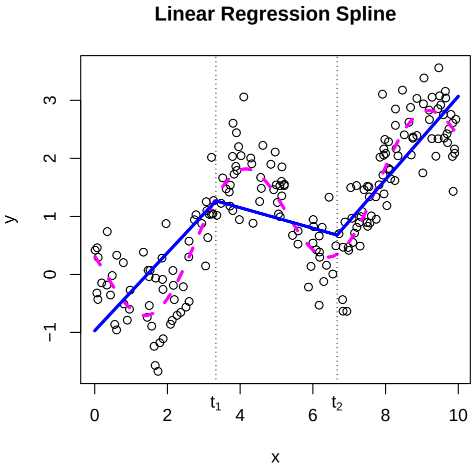
and

$$x_{i3} = \begin{cases} 0 & \text{for } x_i \leq t_2 \\ x_i - t_2 & \text{for } x_i > t_2 \end{cases}$$

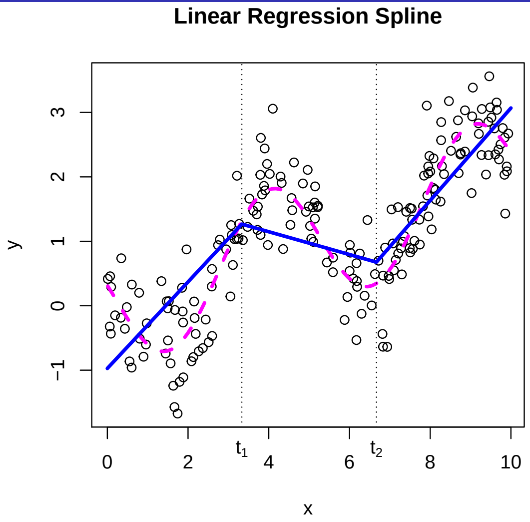


- In R, we could code the *basis* (i.e., the regressors) for the linear spline as follows:

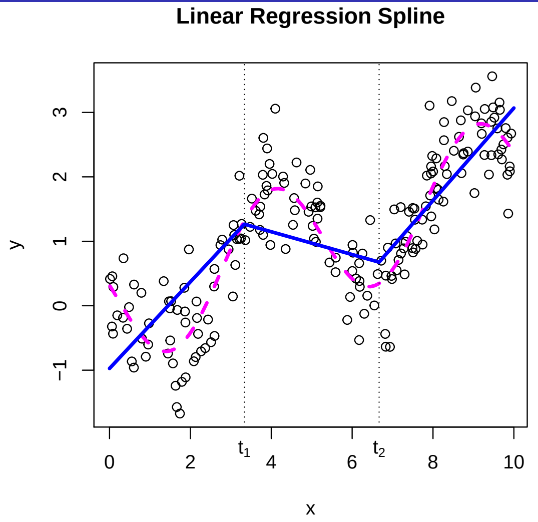
```
> x1 <- x # define spline regressors
> x2 <- (x - t[1]) * (x > t[1])
> x3 <- (x - t[2]) * (x > t[2])
> linear.spline <- lm(y ~ x1 + x2 + x3)
> brief(linear.spline)

           (Intercept)     x1     x2
Estimate        -0.973 0.6718 -0.849
Std. Error       0.169 0.0714  0.113
              x3
Estimate   0.895
Std. Error 0.107

 Residual SD = 0.694 on 196 df, R-squared = 0.643
```



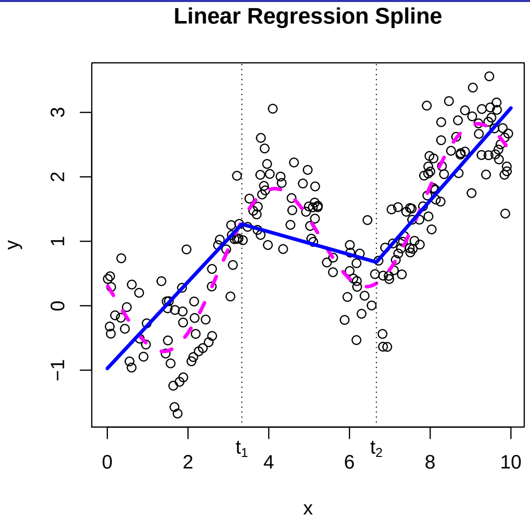
- The coefficients have straightforward interpretations:



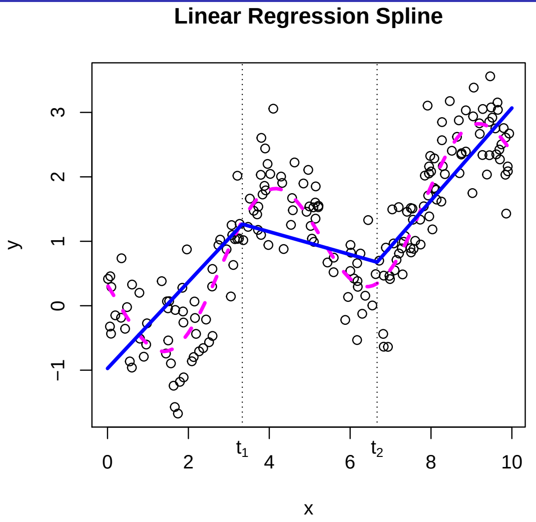
- The coefficients have straightforward interpretations:
  - $\widehat{\beta}_0 = -0.973$ is the estimated expectation of the response y when $x = 0$.



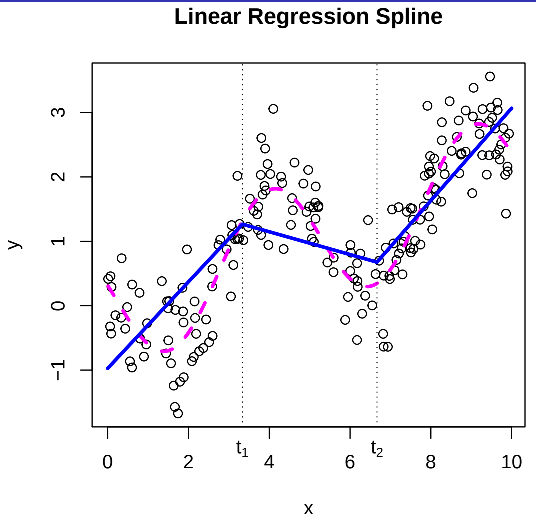
- The coefficients have straightforward interpretations:
  - $\widehat{\beta}_0 = -0.973$ is the estimated expectation of the response y when $x = 0$.
  - $\widehat{\beta}_1 = 0.672$ is the estimated slope in the first interval.



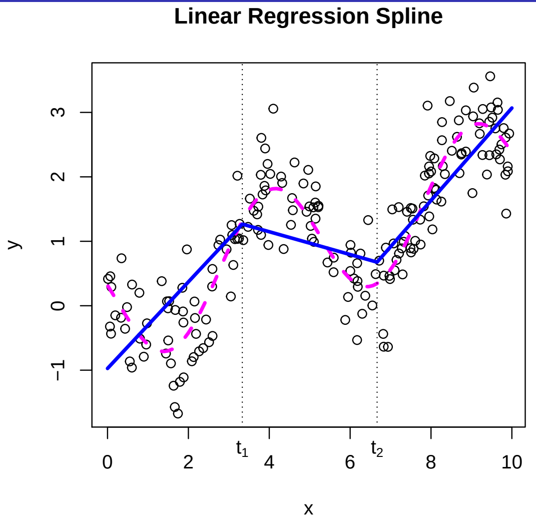
- The coefficients have straightforward interpretations:
  - $\widehat{\beta}_0 = -0.973$ is the estimated expectation of the response y when $x = 0$.
  - $\widehat{\beta}_1 = 0.672$ is the estimated slope in the first interval.
  - $\widehat{\beta}_2 = -0.849$ is the estimated *change* in slope between the first and second interval



- The coefficients have straightforward interpretations:
  - $\widehat{\beta}_0 = -0.973$ is the estimated expectation of the response y when $x = 0$.
  - $\widehat{\beta}_1 = 0.672$ is the estimated slope in the first interval.
  - $\widehat{\beta}_2 = -0.849$ is the estimated *change* in slope between the first and second interval
  - $\widehat{\beta}_3 = 0.895$ is the estimated *change* in slope between the second and third interval.



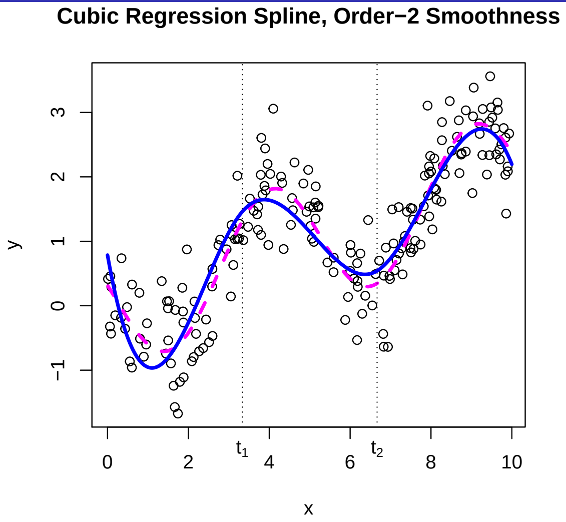
- The coefficients have straightforward interpretations:
  - $\widehat{\beta}_0 = -0.973$ is the estimated expectation of the response y when $x = 0$.
  - $\widehat{\beta}_1 = 0.672$ is the estimated slope in the first interval.
  - $\widehat{\beta}_2 = -0.849$ is the estimated *change* in slope between the first and second interval
  - $\widehat{\beta}_3 = 0.895$ is the estimated *change* in slope between the second and third interval.
- Our goal is to maintain interpretability even in much more complex spline models.



Regression Spline Basics

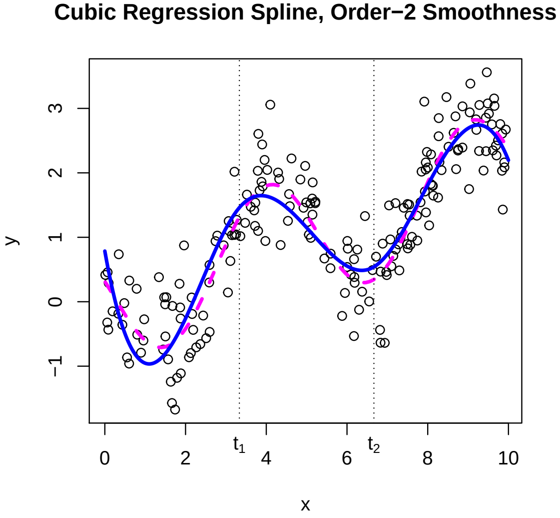
Cubic Regression Spline with Order-2 Smoothness

- This approach generalizes readily to higher-degree polyomials, the most common of which is the third-degree or *cubic regression spline*.



- This approach generalizes readily to higher-degree polyomials, the most common of which is the third-degree or *cubic regression spline*.
- Removing both the linear terms ($\beta_{21}x_{i2}$ and $\beta_{31}x_{i3}$) and the quadratic terms ($\beta_{22}x_{i2}^2$ and $\beta_{32}x_{i3}^2$) from the the model forces the slope and curvature to be equal on both sides of each knot, producing *order-2 smoothess* and a traditional cubic regression spline:

$$y_i = \beta_0 + \beta_{11}x_{i1} + \beta_{12}x_{i1}^2 + \beta_{13}x_{i1}^3 + \beta_{23}x_{i2}^3 + \beta_{33}x_{i3}^3 + \varepsilon_i$$



Using `gspline()` for Fitting Generalized Regression Spline Models

- The `gspline()` (generalized regression `spline`) function in the **carEx** package can fit all of these piecewise polynomial and regression spline models.

Using `gspline()` for Fitting Generalized Regression Spline Models

- The `gspline()` (generalized regression `spline`) function in the **carEx** package can fit all of these piecewise polynomial and regression spline models.
- `gspline()` has substantial additional flexibility.

Using `gspline()` for Fitting Generalized Regression Spline Models

- The `gspline()` (generalized regression `spline`) function in the **carEx** package can fit all of these piecewise polynomial and regression spline models.
- `gspline()` has substantial additional flexibility.
- We'll focus here on three arguments to `gspline()` that allow us to reproduce all of the preceding examples:

Using `gspline()` for Fitting Generalized Regression Spline Models

- The `gspline()` (generalized regression `spline`) function in the **carEx** package can fit all of these piecewise polynomial and regression spline models.
- `gspline()` has substantial additional flexibility.
- We'll focus here on three arguments to `gspline()` that allow us to reproduce all of the preceding examples:
  - `knots`: a vector giving the locations of the knots, with no default.

Using `gspline()` for Fitting Generalized Regression Spline Models

- The `gspline()` (generalized regression `spline`) function in the **carEx** package can fit all of these piecewise polynomial and regression spline models.
- `gspline()` has substantial additional flexibility.
- We'll focus here on three arguments to `gspline()` that allow us to reproduce all of the preceding examples:
  - `knots`: a vector giving the locations of the knots, with no default.
  - `degree`: the degree of the regression spline, defaulting to 3 (a cubic spline); can be a vector of length equal to the number of knots plus one (i.e., the number of intervals between the extremes and knots).

Using gspline() for Fitting Generalized Regression Spline Models

- The `gspline()` (generalized regression `spline`) function in the **carEx** package can fit all of these piecewise polynomial and regression spline models.
- `gspline()` has substantial additional flexibility.
- We'll focus here on three arguments to `gspline()` that allow us to reproduce all of the preceding examples:
  - `knots`: a vector giving the locations of the knots, with no default.
  - `degree`: the degree of the regression spline, defaulting to 3 (a cubic spline); can be a vector of length equal to the number of knots plus one (i.e., the number of intervals between the extremes and knots).
  - `smoothness`: the order of smoothness (number of derivatives to be matched) at the knots, defaulting (in simple cases) to one less than `degree`; can also be a vector specifying the smoothness at each knot; `-1` specifies discontinuity.

Using gspline() for Fitting Generalized Regression Spline Models

- The `gspline()` (generalized regression `spline`) function in the **carEx** package can fit all of these piecewise polynomial and regression spline models.
- `gspline()` has substantial additional flexibility.
- We'll focus here on three arguments to `gspline()` that allow us to reproduce all of the preceding examples:
    - `knots`: a vector giving the locations of the knots, with no default.
    - `degree`: the degree of the regression spline, defaulting to 3 (a cubic spline); can be a vector of length equal to the number of knots plus one (i.e., the number of intervals between the extremes and knots).
    - `smoothness`: the order of smoothness (number of derivatives to be matched) at the knots, defaulting (in simple cases) to one less than `degree`; can also be a vector specifying the smoothness at each knot; `-1` specifies discontinuity.
- `gspline()` returns a function (more technically, a *closure*) that has several arguments:

Using `gspline()` for Fitting Generalized Regression Spline Models

- The `gspline()` (generalized regression `spline`) function in the **carEx** package can fit all of these piecewise polynomial and regression spline models.
- `gspline()` has substantial additional flexibility.
- We'll focus here on three arguments to `gspline()` that allow us to reproduce all of the preceding examples:
  - `knots`: a vector giving the locations of the knots, with no default.
  - `degree`: the degree of the regression spline, defaulting to 3 (a cubic spline); can be a vector of length equal to the number of knots plus one (i.e., the number of intervals between the extremes and knots).
  - `smoothness`: the order of smoothness (number of derivatives to be matched) at the knots, defaulting (in simple cases) to one less than `degree`; can also be a vector specifying the smoothness at each knot; `-1` specifies discontinuity.
- `gspline()` returns a function (more technically, a *closure*) that has several arguments:
  - Its first argument, `x`, is normally the predictor vector.

Using `gspline()` for Fitting Generalized Regression Spline Models

- The `gspline()` (generalized regression `spline`) function in the **carEx** package can fit all of these piecewise polynomial and regression spline models.
- `gspline()` has substantial additional flexibility.
- We'll focus here on three arguments to `gspline()` that allow us to reproduce all of the preceding examples:
  - `knots`: a vector giving the locations of the knots, with no default.
  - `degree`: the degree of the regression spline, defaulting to 3 (a cubic spline); can be a vector of length equal to the number of knots plus one (i.e., the number of intervals between the extremes and knots).
  - `smoothness`: the order of smoothness (number of derivatives to be matched) at the knots, defaulting (in simple cases) to one less than `degree`; can also be a vector specifying the smoothness at each knot; `-1` specifies discontinuity.
- `gspline()` returns a function (more technically, a *closure*) that has several arguments:
  - Its first argument, `x`, is normally the predictor vector.
  - The function returned by `gspline()` can also be used to generate hypothesis matrices, a topic we don't pursue here.

- Here are all of the preceding examples (and more) fit via `gspline()`; in each case, we'd use the generated spline function in a call of the form `lm(y ~ sp(x))`, and the knots are in the vector `t <- c(10/3, 20/3)`:

Using `gspline()` for Fitting Generalized Regression Spline Models
Simple Examples

- Here are all of the preceding examples (and more) fit via `gspline()`; in each case, we'd use the generated spline function in a call of the form `lm(y ~ sp(x))`, and the knots are in the vector `t <- c(10/3, 20/3)`:
  - piecewise constant fit, `sp <- gspline(knots=t, degree=0, smoothness=-1)`.

- Here are all of the preceding examples (and more) fit via `gspline()`; in each case, we'd use the generated spline function in a call of the form `lm(y ~ sp(x))`, and the knots are in the vector `t <- c(10/3, 20/3)`:
  - piecewise constant fit, `sp <- gspline(knots=t, degree=0, smoothness=-1)`.
  - piecewise linear fit, `sp <- gspline(knots=t, degree=1, smoothness=-1)`

- Here are all of the preceding examples (and more) fit via `gspline()`; in each case, we'd use the generated spline function in a call of the form `lm(y ~ sp(x))`, and the knots are in the vector `t <- c(10/3, 20/3)`:
  - piecewise constant fit, `sp <- gspline(knots=t, degree=0, smoothness=-1)`.
  - piecewise linear fit, `sp <- gspline(knots=t, degree=1, smoothness=-1)`
  - linear regression spline, `sp <- gspline(knots=t, degree=1, smoothness=0)` or just `sp <- gspline(knots=t, degree=1)`

- Here are all of the preceding examples (and more) fit via `gspline()`; in each case, we'd use the generated spline function in a call of the form `lm(y ~ sp(x))`, and the knots are in the vector `t <- c(10/3, 20/3)`:
  - piecewise constant fit, `sp <- gspline(knots=t, degree=0, smoothness=-1)`.
  - piecewise linear fit, `sp <- gspline(knots=t, degree=1, smoothness=-1)`
  - linear regression spline, `sp <- gspline(knots=t, degree=1, smoothness=0)` or just `sp <- gspline(knots=t, degree=1)`
  - piecewise cubic fit, `sp <- gspline(knots=t, degree=3, smoothness=-1)`

- Here are all of the preceding examples (and more) fit via `gspline()`; in each case, we'd use the generated spline function in a call of the form `lm(y ~ sp(x))`, and the knots are in the vector `t <- c(10/3, 20/3)`:
  - piecewise constant fit, `sp <- gspline(knots=t, degree=0, smoothness=-1)`.
  - piecewise linear fit, `sp <- gspline(knots=t, degree=1, smoothness=-1)`
  - linear regression spline, `sp <- gspline(knots=t, degree=1, smoothness=0)` or just `sp <- gspline(knots=t, degree=1)`
  - piecewise cubic fit, `sp <- gspline(knots=t, degree=3, smoothness=-1)`
  - cubic regression spline with continuity only, `sp <- gspline(knots=t, degree=3, smoothness=0)`

- Here are all of the preceding examples (and more) fit via `gspline()`; in each case, we'd use the generated spline function in a call of the form `lm(y ~ sp(x))`, and the knots are in the vector `t <- c(10/3, 20/3)`:
  - piecewise constant fit, `sp <- gspline(knots=t, degree=0, smoothness=-1)`.
  - piecewise linear fit, `sp <- gspline(knots=t, degree=1, smoothness=-1)`
  - linear regression spline, `sp <- gspline(knots=t, degree=1, smoothness=0)` or just `sp <- gspline(knots=t, degree=1)`
  - piecewise cubic fit, `sp <- gspline(knots=t, degree=3, smoothness=-1)`
  - cubic regression spline with continuity only, `sp <- gspline(knots=t, degree=3, smoothness=0)`
  - cubic regression spline with matching slopes, `sp <- gspline(knots=t, degree=3, smoothness=1)`

- Here are all of the preceding examples (and more) fit via `gspline()`; in each case, we'd use the generated spline function in a call of the form `lm(y ~ sp(x))`, and the knots are in the vector `t <- c(10/3, 20/3)`:
  - piecewise constant fit, `sp <- gspline(knots=t, degree=0, smoothness=-1)`.
  - piecewise linear fit, `sp <- gspline(knots=t, degree=1, smoothness=-1)`
  - linear regression spline, `sp <- gspline(knots=t, degree=1, smoothness=0)` or just `sp <- gspline(knots=t, degree=1)`
  - piecewise cubic fit, `sp <- gspline(knots=t, degree=3, smoothness=-1)`
  - cubic regression spline with continuity only, `sp <- gspline(knots=t, degree=3, smoothness=0)`
  - cubic regression spline with matching slopes, `sp <- gspline(knots=t, degree=3, smoothness=1)`
  - cubic regression spline with matching slopes and curvature, `sp <- gspline(knots=t, degree=3, smoothness=2)` or just `sp <- gspline(knots=t)`

- *Natural splines* are B-splines with additional *boundary knots*, typically placed at the extremes of x, and the additional constraints that the fitted regression is linear beyond the boundary knots.

- *Natural splines* are B-splines with additional *boundary knots*, typically placed at the extremes of x, and the additional constraints that the fitted regression is linear beyond the boundary knots.
  - Derivatives up to order one less than the degree of the interior spline polynomials also match at the boundaries.

- *Natural splines* are B-splines with additional *boundary knots*, typically placed at the extremes of x, and the additional constraints that the fitted regression is linear beyond the boundary knots.
    - Derivatives up to order one less than the degree of the interior spline polynomials also match at the boundaries.
    - Natural splines tend to produce less wild extrapolations than B-splines and may also behave more reasonably in the interior near the boundaries.

- *Natural splines* are B-splines with additional *boundary knots*, typically placed at the extremes of x, and the additional constraints that the fitted regression is linear beyond the boundary knots.
  - Derivatives up to order one less than the degree of the interior spline polynomials also match at the boundaries.
  - Natural splines tend to produce less wild extrapolations than B-splines and may also behave more reasonably in the interior near the boundaries.
  - Cubic natural splines are implemented by the `ns()` function in the standard R **splines** package, and by the `rcs()` and related functions in the **Hmisc** package (Harrell, 2015).

- *Natural splines* are B-splines with additional *boundary knots*, typically placed at the extremes of x, and the additional constraints that the fitted regression is linear beyond the boundary knots.
    - Derivatives up to order one less than the degree of the interior spline polynomials also match at the boundaries.
    - Natural splines tend to produce less wild extrapolations than B-splines and may also behave more reasonably in the interior near the boundaries.
    - Cubic natural splines are implemented by the `ns()` function in the standard R **splines** package, and by the `rcs()` and related functions in the **Hmisc** package (Harrell, 2015).
- `gspline()` doesn't treat boundary knots specially, but using the `knots`, `degree`, and `smoothness` arguments it's simple to specify the equivalent of a natural spline: e.g.,

```
gspline(knots=c(0, 10/3, 20/3, 10),
        degree=c(1, 3, 3, 3, 1),
        smoothness=c(2, 2, 2, 2))
```

Generalized Regression Splines: Theory

Notation for Generalized Regression Splines

- The *generalized regression splines* implemented in the **carEx** package are piecewise polynomial functions on $k+1$ intervals formed by k knots partitioning the real line. $(-\infty, t_1], (t_1, t_2], \ldots, (t_{i-1}, t_i], \ldots, (t_k, \infty)$.

- The *generalized regression splines* implemented in the **carEx** package are piecewise polynomial functions on $k+1$ intervals formed by k knots partitioning the real line. $(-\infty, t_1], (t_1, t_2], \ldots, (t_{i-1}, t_i], \ldots, (t_k, \infty)$.
- Such a spline is parametrized by three vectors:

- The *generalized regression splines* implemented in the **carEx** package are piecewise polynomial functions on $k+1$ intervals formed by k knots partitioning the real line. $(-\infty, t_1], (t_1, t_2], \ldots, (t_{i-1}, t_i], \ldots, (t_k, \infty)$.
- Such a spline is parametrized by three vectors:
  1. a vector of *knots*, $t_1 < t_2 < \ldots < t_k$, of length $k > 0$,

- The *generalized regression splines* implemented in the **carEx** package are piecewise polynomial functions on $k+1$ intervals formed by k knots partitioning the real line. $(-\infty, t_1], (t_1, t_2], \ldots, (t_{i-1}, t_i], \ldots, (t_k, \infty)$.
- Such a spline is parametrized by three vectors:
  1. a vector of *knots*, $t_1 < t_2 < ... < t_k$, of length $k > 0$,
  2. a vector of polynomial *degrees*, $d_1, d_2, ..., d_{k+1}$, of length $k+1$,

- The *generalized regression splines* implemented in the **carEx** package are piecewise polynomial functions on $k + 1$ intervals formed by k knots partitioning the real line. $(-\infty, t_1], (t_1, t_2], \ldots, (t_{i-1}, t_i], \ldots, (t_k, \infty)$.
- Such a spline is parametrized by three vectors:
  1. a vector of *knots*, $t_1 < t_2 < ... < t_k$, of length $k > 0$,
  2. a vector of polynomial *degrees*, $d_1, d_2, ..., d_{k+1}$, of length $k + 1$,
  3. a vector of orders of continuity or *smoothness*, $c_1, c_2, ..., c_k$, of length k, the highest order for which the derivatives on each side of a knot t_i match.

- Let X_f be an $n \times q$ matrix for a model whose coefficients are subject to c linearly independent constraints given by a $c \times q$ matrix C.

- Let X_f be an $n \times q$ matrix for a model whose coefficients are subject to c linearly independent constraints given by a $c \times q$ matrix C.
- The linear space for the model is $\mathcal{M} = \{\eta = X_f \phi : \phi \in \mathbb{R}^q, C\phi = 0\}$.

- Let X_f be an $n \times q$ matrix for a model whose coefficients are subject to c linearly independent constraints given by a $c \times q$ matrix C.
- The linear space for the model is $\mathcal{M} = \{\eta = X_f \phi : \phi \in \mathbb{R}^q, C\phi = 0\}$.
- We wish to construct an $n \times p$ model matrix X with $p = q - c$ so that $\mathcal{M} = \{\eta = X\beta : \beta \in \mathbb{R}^p\}$.

Generalized Regression Splines: Theory

General Principles

- Let X_f be an $n \times q$ matrix for a model whose coefficients are subject to c linearly independent constraints given by a $c \times q$ matrix C.
- The linear space for the model is $\mathcal{M} = \{\eta = X_f\phi : \phi \in \mathbb{R}^q, C\phi = 0\}$.
- We wish to construct an $n \times p$ model matrix X with $p = q - c$ so that $\mathcal{M} = \{\eta = X\beta : \beta \in \mathbb{R}^p\}$.
- We further want the parameters β to provide p specified linearly independent functions of ϕ represented by the rows of the $p \times q$ matrix E whose rows are linearly independent of the rows of C to ensure that they are not equal to 0 on $\mathcal{M}$.

- Let X_f be an $n \times q$ matrix for a model whose coefficients are subject to c linearly independent constraints given by a $c \times q$ matrix C.
- The linear space for the model is $\mathcal{M} = \{\eta = X_f \phi : \phi \in \mathbb{R}^q, C\phi = 0\}$.
- We wish to construct an $n \times p$ model matrix X with $p = q - c$ so that $\mathcal{M} = \{\eta = X\beta : \beta \in \mathbb{R}^p\}$.
- We further want the parameters β to provide p specified linearly independent functions of ϕ represented by the rows of the $p \times q$ matrix E whose rows are linearly independent of the rows of C to ensure that they are not equal to 0 on $\mathcal{M}$.
- Then the $q \times q$ partitioned matrix $\begin{bmatrix} C \\ E \end{bmatrix}$ has linearly independent rows and is invertible with a conformably partitioned inverse $\begin{bmatrix} F & G \end{bmatrix} = \begin{bmatrix} C \\ E \end{bmatrix}^{-1}$.

- Consider the model matrix $X = X_f G$.

Generalized Regression Splines: Theory

General Principles

- Consider the model matrix $X = X_f G$.
  - It can be shown that $\mathcal{M} = \{X\beta : \beta \in \mathbb{R}^p\}$ and that for any $\phi \in \mathbb{R}^q$, such that $C\phi = 0$, $\beta = E\phi$.

- Consider the model matrix $X = X_f G$.
  - It can be shown that $\mathcal{M} = \{X\beta : \beta \in \mathbb{R}^p\}$ and that for any $\phi \in \mathbb{R}^q$, such that $C\phi = 0$, $\beta = E\phi$.
  - If X is of full rank, this defines a one–one correspondence between $\beta \in \mathbb{R}^p$ and $\{\phi \in \mathbb{R}^q : C\phi = 0\}$ given by $\beta = E\phi$ and $\phi = G\beta$.

Generalized Regression Splines: Theory

General Principles

- Consider the model matrix $X = X_f G$.
  - It can be shown that $\mathcal{M} = \{X\beta : \beta \in \mathbb{R}^p\}$ and that for any $\phi \in \mathbb{R}^q$, such that $C\phi = 0$, $\beta = E\phi$.
  - If X is of full rank, this defines a one–one correspondence between $\beta \in \mathbb{R}^p$ and $\{\phi \in \mathbb{R}^q : C\phi = 0\}$ given by $\beta = E\phi$ and $\phi = G\beta$.
- Given the least-squares estimator $\widehat{\beta}$ of β, We can estimate any linear function $\psi = L\phi$ of ϕ under the constraint $C\phi = 0$ with the estimator $\widehat{\psi} = A\widehat{\beta}$ with $A = LG$.

- Consider the model matrix $X = X_f G$.
    - It can be shown that $\mathcal{M} = \{X\beta : \beta \in \mathbb{R}^p\}$ and that for any $\phi \in \mathbb{R}^q$, such that $C\phi = 0$, $\beta = E\phi$.
    - If X is of full rank, this defines a one–one correspondence between $\beta \in \mathbb{R}^p$ and $\{\phi \in \mathbb{R}^q : C\phi = 0\}$ given by $\beta = E\phi$ and $\phi = G\beta$.
- Given the least-squares estimator $\widehat{\beta}$ of β, We can estimate any linear function $\psi = L\phi$ of ϕ under the constraint $C\phi = 0$ with the estimator $\widehat{\psi} = A\widehat{\beta}$ with $A = LG$.
    - Thus, G serves as a post-multiplier to transform X_f into a model matrix $X = X_f G$ that can be used in a linear model.

- Consider the model matrix $X = X_f G$.
  - It can be shown that $\mathcal{M} = \{X\beta : \beta \in \mathbb{R}^p\}$ and that for any $\phi \in \mathbb{R}^q$, such that $C\phi = 0$, $\beta = E\phi$.
  - If X is of full rank, this defines a one–one correspondence between $\beta \in \mathbb{R}^p$ and $\{\phi \in \mathbb{R}^q : C\phi = 0\}$ given by $\beta = E\phi$ and $\phi = G\beta$.
- Given the least-squares estimator $\widehat{\beta}$ of β, We can estimate any linear function $\psi = L\phi$ of ϕ under the constraint $C\phi = 0$ with the estimator $\widehat{\psi} = A\widehat{\beta}$ with $A = LG$.
  - Thus, G serves as a post-multiplier to transform X_f into a model matrix $X = X_f G$ that can be used in a linear model.
  - G also serves as a post-multiplier to transform any general linear hypothesis matrix expressed in terms of ϕ into a general linear hypothesis matrix in terms of of β.

Generalized Regression Splines: Theory

- Our goal is to generate model matrices for splines that produces interpretable coefficients and simple estimates and tests of properties of the spline that are linear functions of parameters: slope, curvature, discontinuities, etc.

Generalized Regression Splines: Theory

- Our goal is to generate model matrices for splines that produces interpretable coefficients and simple estimates and tests of properties of the spline that are linear functions of parameters: slope, curvature, discontinuities, etc.
- Generating a model matrix for a piecewise polynomial is sometimes simple, as in our introductory examples, but that isn't true generally.

Generalized Regression Splines: Theory

Application to Splines

- Our goal is to generate model matrices for splines that produces interpretable coefficients and simple estimates and tests of properties of the spline that are linear functions of parameters: slope, curvature, discontinuities, etc.
- Generating a model matrix for a piecewise polynomial is sometimes simple, as in our introductory examples, but that isn't true generally.
- Generating model matrices in more general situations, for example with degrees that are not monotone, nor monotone increasing as the index radiates from a central value, is more challenging.

- Our goal is to generate model matrices for splines that produces interpretable coefficients and simple estimates and tests of properties of the spline that are linear functions of parameters: slope, curvature, discontinuities, etc.
- Generating a model matrix for a piecewise polynomial is sometimes simple, as in our introductory examples, but that isn't true generally.
- Generating model matrices in more general situations, for example with degrees that are not monotone, nor monotone increasing as the index radiates from a central value, is more challenging.
- The approach described here works for any pattern of degrees, d_i and smoothness constraints, c_i.

- Our goal is to generate model matrices for splines that produces interpretable coefficients and simple estimates and tests of properties of the spline that are linear functions of parameters: slope, curvature, discontinuities, etc.
- Generating a model matrix for a piecewise polynomial is sometimes simple, as in our introductory examples, but that isn't true generally.
- Generating model matrices in more general situations, for example with degrees that are not monotone, nor monotone increasing as the index radiates from a central value, is more challenging.
- The approach described here works for any pattern of degrees, d_i and smoothness constraints, c_i.
- We start by constucting a matrix, X_f, for a spline in which the polynomial degree in each interval is the maximal value, $\max(d_i)$, and then construct constraints for the coefficients of this model to produce the desired spline.

- Extending our earlier examples, consider a spline, $\mathcal{S}$, with knots at $t = (10/3, 20/3)$, polynomial degrees, $d = (2, 3, 2)$, and smoothness, $c = (1, 2)$.

- Extending our earlier examples, consider a spline, $\mathcal{S}$, with knots at $t = (10/3, 20/3)$, polynomial degrees, $d = (2, 3, 2)$, and smoothness, $c = (1, 2)$.
- Columns of the full matrix X_f contain the intercept, and linear, quadratic, and cubic terms in each interval of the spline.

Generalized Regression Splines: Theory

Application to Splines

- Extending our earlier examples, consider a spline, $\mathcal{S}$, with knots at $t = (10/3, 20/3)$, polynomial degrees, $d = (2, 3, 2)$, and smoothness, $c = (1, 2)$.
- Columns of the full matrix X_f contain the intercept, and linear, quadratic, and cubic terms in each interval of the spline.
- To create an instance of X_f we need to specify the values over which the matrix is evaluated, say at $x = 0, 1, \ldots, 10$ (where the function `Xf()`, used to generate the matrix, is local to the `gspline()` function in the **carEx** package and thus not normally called by the user):

```
> Xf(0:10, knots=c(10/3, 20/3), degree = 3)

      X0 X1 X2 X3 X0 X1 X2  X3 X0 X1  X2   X3
f(0)   1  0  0  0  0  0  0   0  0  0   0    0
f(1)   1  1  1  1  0  0  0   0  0  0   0    0
f(2)   1  2  4  8  0  0  0   0  0  0   0    0
f(3)   1  3  9 27  0  0  0   0  0  0   0    0
f(4)   0  0  0  0  1  4 16  64  0  0   0    0
f(5)   0  0  0  0  1  5 25 125  0  0   0    0
f(6)   0  0  0  0  1  6 36 216  0  0   0    0
f(7)   0  0  0  0  0  0  0   0  1  7  49  343
f(8)   0  0  0  0  0  0  0   0  1  8  64  512
f(9)   0  0  0  0  0  0  0   0  1  9  81  729
f(10)  0  0  0  0  0  0  0   0  1 10 100 1000
```

- The model for the unconstrained maximal polynomial is $X_f \phi : \phi \in \mathbb{R}^{12}$.

- The model for the unconstrained maximal polynomial is $X_f\phi : \phi \in \mathbb{R}^{12}$.
- We impose three types of constraints on ϕ.

- The model for the unconstrained maximal polynomial is $X_f \phi : \phi \in \mathbb{R}^{12}$.
- We impose three types of constraints on ϕ.
    1. $X_f \phi$ should evaluate to 0 at $x = 0$ so an intercept term in the model will have the correct interpretation.

- The model for the unconstrained maximal polynomial is $X_f\phi : \phi \in \mathbb{R}^{12}$.
- We impose three types of constraints on ϕ.
  1. $X_f\phi$ should evaluate to 0 at $x = 0$ so an intercept term in the model will have the correct interpretation.
  2. The limits of the value and of the first derivative of the spline must be the same when approaching the first knot from the right or from the left, and the limits of the value, the first and second derivatives should be the same when approaching the second knot from the right or from the left.

- The model for the unconstrained maximal polynomial is $X_f\phi : \phi \in \mathbb{R}^{12}$.
- We impose three types of constraints on ϕ.
  1. $X_f\phi$ should evaluate to 0 at $x = 0$ so an intercept term in the model will have the correct interpretation.
  2. The limits of the value and of the first derivative of the spline must be the same when approaching the first knot from the right or from the left, and the limits of the value, the first and second derivatives should be the same when approaching the second knot from the right or from the left.
  3. The degree of the polynomial in the first and third intervals must be reduced to 2.

- The constraint matrix, C, is created by the (internal) `Cmat()` function:

```
> Cmat(knots=c(10/3, 20/3), degree=c(2, 3, 2), smooth=c(1, 2))
         X0      X1       X2      X3 X0      X1       X2       X3 X0     X1     X2     X3
f(0)      1  0.0000   0.0000   0.000  0  0.0000   0.0000    0.000  0 0.0000  0.000   0.00
C0|3.33  -1 -3.3333 -11.1111 -37.037  1  3.3333  11.1111   37.037  0 0.0000  0.000   0.00
C1|3.33   0 -1.0000  -6.6667 -33.333  0  1.0000   6.6667   33.333  0 0.0000  0.000   0.00
C0|6.67   0  0.0000   0.0000   0.000 -1 -6.6667 -44.4444 -296.296  1 6.6667 44.444 296.30
C1|6.67   0  0.0000   0.0000   0.000  0 -1.0000 -13.3333 -133.333  0 1.0000 13.333 133.33
C2|6.67   0  0.0000   0.0000   0.000  0  0.0000  -2.0000  -40.000  0 0.0000  2.000  40.00
I.1.3     0  0.0000   0.0000   1.000  0  0.0000   0.0000    0.000  0 0.0000  0.000   0.00
I.3.3     0  0.0000   0.0000   0.000  0  0.0000   0.0000    0.000  0 0.0000  0.000   1.00
attr(,"ranks")
  npar.full         C.n      C.rank spline.rank
         12           8           8           4
attr(,"d")
[1] 469.012615  63.081291  10.489239   3.528970   0.982249   0.816848   0.375998   0.070072
```

- The constraint matrix, C, is created by the (internal) `Cmat()` function:

```
> Cmat(knots=c(10/3, 20/3), degree=c(2, 3, 2), smooth=c(1, 2))
          X0      X1       X2      X3 X0      X1       X2       X3 X0     X1     X2     X3
f(0)       1  0.0000   0.0000   0.000  0  0.0000   0.0000    0.000  0 0.0000  0.000   0.00
C0|3.33   -1 -3.3333 -11.1111 -37.037  1  3.3333  11.1111   37.037  0 0.0000  0.000   0.00
C1|3.33    0 -1.0000  -6.6667 -33.333  0  1.0000   6.6667   33.333  0 0.0000  0.000   0.00
C0|6.67    0  0.0000   0.0000   0.000 -1 -6.6667 -44.4444 -296.296  1 6.6667 44.444 296.30
C1|6.67    0  0.0000   0.0000   0.000  0 -1.0000 -13.3333 -133.333  0 1.0000 13.333 133.33
C2|6.67    0  0.0000   0.0000   0.000  0  0.0000  -2.0000  -40.000  0 0.0000  2.000  40.00
I.1.3      0  0.0000   0.0000   1.000  0  0.0000   0.0000    0.000  0 0.0000  0.000   0.00
I.3.3      0  0.0000   0.0000   0.000  0  0.0000   0.0000    0.000  0 0.0000  0.000   1.00
attr(,"ranks")
  npar.full         C.n      C.rank spline.rank
         12           8           8           4
attr(,"d")
[1] 469.012615  63.081291  10.489239   3.528970   0.982249   0.816848   0.375998   0.070072
```

- The row labels of the constraint matrix show the role of each row.

- The constraint matrix, C, is created by the (internal) `Cmat()` function:

```
> Cmat(knots=c(10/3, 20/3), degree=c(2, 3, 2), smooth=c(1, 2))

          X0      X1       X2      X3 X0      X1       X2       X3 X0     X1     X2     X3
f(0)       1  0.0000   0.0000   0.000  0  0.0000   0.0000    0.000  0 0.0000  0.000   0.00
C0|3.33   -1 -3.3333 -11.1111 -37.037  1  3.3333  11.1111   37.037  0 0.0000  0.000   0.00
C1|3.33    0 -1.0000  -6.6667 -33.333  0  1.0000   6.6667   33.333  0 0.0000  0.000   0.00
C0|6.67    0  0.0000   0.0000   0.000 -1 -6.6667 -44.4444 -296.296  1 6.6667 44.444 296.30
C1|6.67    0  0.0000   0.0000   0.000  0 -1.0000 -13.3333 -133.333  0 1.0000 13.333 133.33
C2|6.67    0  0.0000   0.0000   0.000  0  0.0000  -2.0000  -40.000  0 0.0000  2.000  40.00
I.1.3      0  0.0000   0.0000   1.000  0  0.0000   0.0000    0.000  0 0.0000  0.000   0.00
I.3.3      0  0.0000   0.0000   0.000  0  0.0000   0.0000    0.000  0 0.0000  0.000   1.00
attr(,"ranks")
  npar.full         C.n      C.rank spline.rank
         12           8           8           4
attr(,"d")
[1] 469.012615  63.081291  10.489239   3.528970   0.982249   0.816848   0.375998   0.070072
```

- The row labels of the constraint matrix show the role of each row.
- The `d` attribute contains the vector of singular values of the constraint matrix.

Generalized Regression Splines: Theory

Application to Splines

- The matrix E of estimable functions is created by the (internal) `Emat()` function:

```
> Emat(knots=c(10/3, 20/3), degree=c(2, 3, 2), smooth=c(1, 2))
        X0 X1 X2  X3 X0 X1 X2 X3 X0 X1 X2 X3
D1|0     0  1  0   0  0  0  0  0  0  0  0  0
D2|0     0  0  2   0  0  0  0  0  0  0  0  0
C2|3.33  0  0 -2 -20  0  0  2 20  0  0  0  0
C3|3.33  0  0  0  -6  0  0  0  6  0  0  0  0
```

- The matrix E of estimable functions is created by the (internal) `Emat()` function:

```
> Emat(knots=c(10/3, 20/3), degree=c(2, 3, 2), smooth=c(1, 2))

         X0 X1 X2  X3 X0 X1 X2 X3 X0 X1 X2 X3
D1|0      0  1  0   0  0  0  0  0  0  0  0  0
D2|0      0  0  2   0  0  0  0  0  0  0  0  0
C2|3.33  0  0 -2 -20  0  0  2 20  0  0  0  0
C3|3.33  0  0  0  -6  0  0  0  6  0  0  0  0
```

- The row labels signify the first derivative at $x = 0$, `D1(0)`, the second derivative at $x = 0$, `D2(0)`, the change in the second derivative at $x = 10/3$, `C(3.33).2`, and the change in the third derivative at $x = 10/3$, `C(3.33).3`.

- The full-rank model matrix $X = X_f G$ for the spline parametrized by linear estimable coefficients is generated as previously described, as a closure produced by the `gspline()` function:

```
> sp <- gspline(knots=c(10/3, 20/3), degree=c(2, 3, 2), smooth=c(1, 2))
> sp(0:10)
      D1|0 D2|0 C2|3.33 C3|3.33
f(0)     0  0.0   0.000   0.000
f(1)     1  0.5   0.000   0.000
f(2)     2  2.0   0.000   0.000
f(3)     3  4.5   0.000   0.000
f(4)     4  8.0   0.222   0.049
f(5)     5 12.5   1.389   0.772
f(6)     6 18.0   3.556   3.160
f(7)     7 24.5   6.722   8.210
f(8)     8 32.0  10.889  16.543
f(9)     9 40.5  16.056  28.210
f(10)   10 50.0  22.222  43.210
```

- The full-rank model matrix $X = X_f G$ for the spline parametrized by linear estimable coefficients is generated as previously described, as a closure produced by the `gspline()` function:

```
> sp <- gspline(knots=c(10/3, 20/3), degree=c(2, 3, 2), smooth=c(1, 2))
> sp(0:10)
      D1|0 D2|0 C2|3.33 C3|3.33
f(0)     0  0.0   0.000   0.000
f(1)     1  0.5   0.000   0.000
f(2)     2  2.0   0.000   0.000
f(3)     3  4.5   0.000   0.000
f(4)     4  8.0   0.222   0.049
f(5)     5 12.5   1.389   0.772
f(6)     6 18.0   3.556   3.160
f(7)     7 24.5   6.722   8.210
f(8)     8 32.0  10.889  16.543
f(9)     9 40.5  16.056  28.210
f(10)   10 50.0  22.222  43.210
```

- The closure `sp()` created by `gspline()` can be used in a linear or similar model formula.

- We use the monthly Canadian unemployment rates from January 1995 to February 2019 to illustrate a model with a discontinuity and, subsequently, a periodic spline component for annual seasonal patterns.

Example: Canadian Unemployment and the 2008 Crash

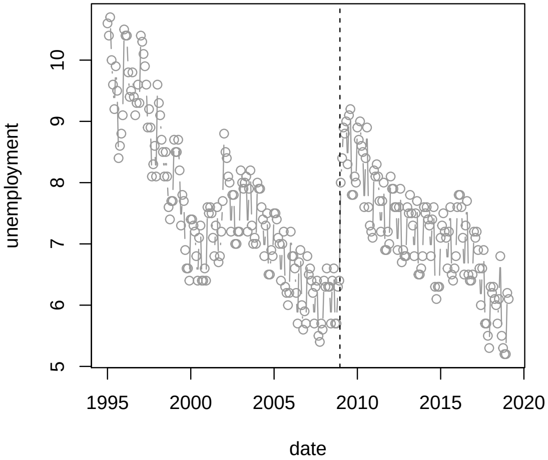
Data

- We use the monthly Canadian unemployment rates from January 1995 to February 2019 to illustrate a model with a discontinuity and, subsequently, a periodic spline component for annual seasonal patterns.
- The data are in the `Unemployment` data set in the **carEx** package:

```
> brief(Unemployment)

290 x 2 data.frame (285 rows omitted)
          date unemployment
           [D]          [n]
1   1995-01-01         10.6
2   1995-02-01         10.4
3   1995-03-01         10.7
. . .
289 2019-01-01          6.2
290 2019-02-01          6.1
```

- For convenience, we add `year` and `month` variables to the data:

```
> toyear <- function(x) (as.numeric(x) -
+     as.numeric(as.Date("2000-01-01")))/365.25
> Unemp <- within(
+   Unemployment,
+   {
+     year <- toyear(date)
+     month <- as.numeric(format(date, "%m"))
+   })
```

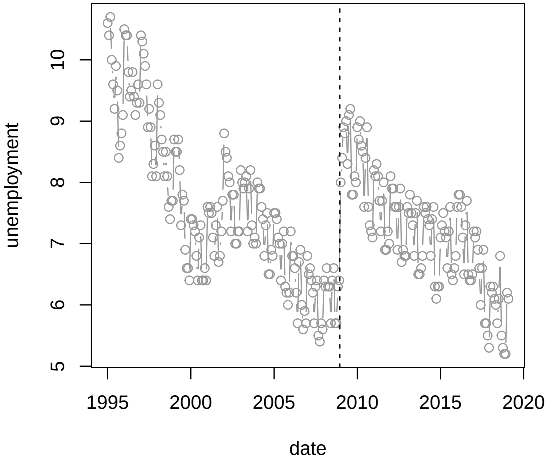


- For convenience, we add `year` and `month` variables to the data:

```
> toyear <- function(x) (as.numeric(x) -
+     as.numeric(as.Date("2000-01-01")))/365.25
> Unemp <- within(
+   Unemployment,
+   {
+     year <- toyear(date)
+     month <- as.numeric(format(date, "%m"))
+   })
```

- Graphing the data, marking the date of the crash, which we take as mid-December:

```
> plot(unemployment ~ date, data=Unemp,
+      col=gray(0.60), type="b")
> abline(v=as.Date("2008-12-15"), lty=2)
```



- We create knots at the quintiles of `year`, and add a knot for the date of the crash:

```
> knots <- quantile(Unemp$year, (1:4)/5)
> knots.d <- sort(c(knots, toyear(as.Date("2008-12-15"))))
```

- We create knots at the quintiles of `year`, and add a knot for the date of the crash:

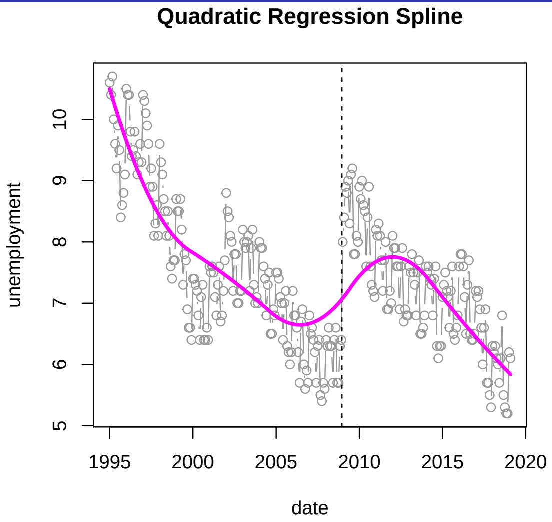
```
> knots <- quantile(Unemp$year, (1:4)/5)
> knots.d <- sort(c(knots, toyear(as.Date("2008-12-15"))))
```

- We proceed to fit several quadratic and cubic regression spline models to the data, with and without a discontinuity at the date of the crash:

```
> sp2 <- gspline(knots=knots, degree=2, smoothness=1)
> sp3 <- gspline(knots=knots, degree=3, smoothness=2)
> sp2.d <- gspline(knots=knots.d, degree=2, smoothness=c(1,1,-1,1,1))
> sp3.d <- gspline(knots=knots.d, degree=3, smoothness=c(2,2,-1,2,2))
> fit2 <- lm(unemployment ~ sp2(year), data=Unemp)
> fit3 <- lm(unemployment ~ sp3(year), data=Unemp)
> fit2.d <- lm(unemployment ~ sp2.d(year), data=Unemp)
> fit3.d <- lm(unemployment ~ sp3.d(year), data=Unemp)
```

- Graphing the various fits:
  - Quadratic regression spline:

```
> plot(unemployment ~ date, data=Unemp,
+       col=gray(0.60), type="b",
+       main="Quadratic Regression Spline")
> abline(v=as.Date("2008-12-15"), lty=2)
> lines(Unemp$date, predict(fit2), lwd=3,
+        col="magenta")
```

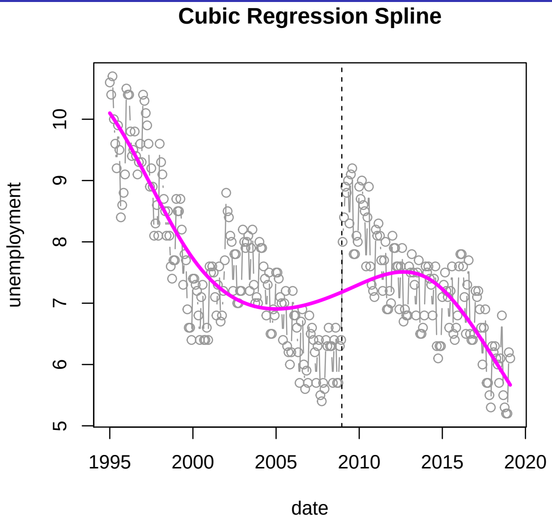


Example: Canadian Unemployment and the 2008 Crash

Regression Spline Models

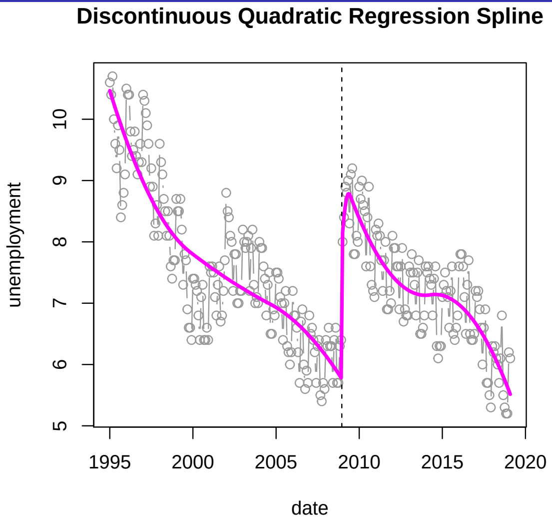
- Graphing the various fits:
  - Cubic regression spline:

```
> plot(unemployment ~ date, data=Unemp,
+       col=gray(0.60), type="b",
+       main="Cubic Regression Spline")
> abline(v=as.Date("2008-12-15"), lty=2)
> lines(Unemp$date, predict(fit3), lwd=3,
+        col="magenta")
```



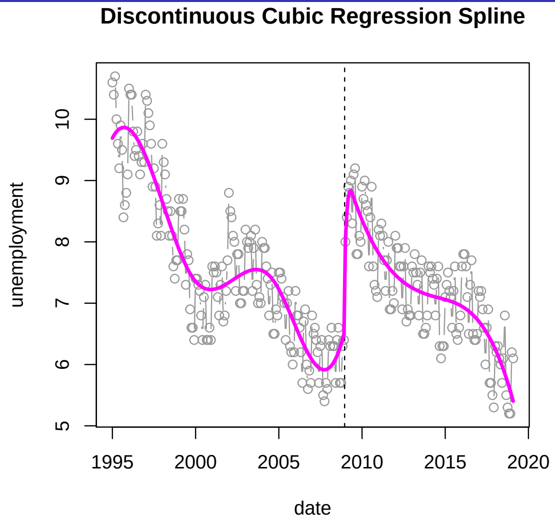
- Graphing the various fits:
  - Quadratic regression spline with discontinuity:

```
> plot(unemployment ~ date, data=Unemp,
+  col=gray(0.60), type="b",
+  main="Discontinuous Quadratic Regression Spline")
> abline(v=as.Date("2008-12-15"), lty=2)
> lines(Unemp$date, predict(fit2.d), lwd=3,
+        col="magenta")
```



- Graphing the various fits:
    - Cubic regression spline with discontinuity:

```
> plot(unemployment ~ date, data=Unemp,
+  col=gray(0.60), type="b",
+  main="Discontinuous Cubic Regression Spline")
> abline(v=as.Date("2008-12-15"), lty=2)
> lines(Unemp$date, predict(fit3.d), lwd=3,
+        col="magenta")
```



- The data have an apparent periodic character not captured by the regression splines models fit so far.

- The data have an apparent periodic character not captured by the regression splines models fit so far.
- To model this pattern, we add a periodic spline component as a function of months, using a cubic spline with period 12 and 4 internal knots at $12 \times (1/5, 2/5, 3/5, 4/5)$ plus a boundary knot at 12.

- The data have an apparent periodic character not captured by the regression splines models fit so far.
- To model this pattern, we add a periodic spline component as a function of months, using a cubic spline with period 12 and 4 internal knots at $12 \times (1/5, 2/5, 3/5, 4/5)$ plus a boundary knot at 12.
- The derivatives parametrizing the periodic spline are derivatives from the left at the maximum knot, which are identified with the same derivatives from the left at 0.

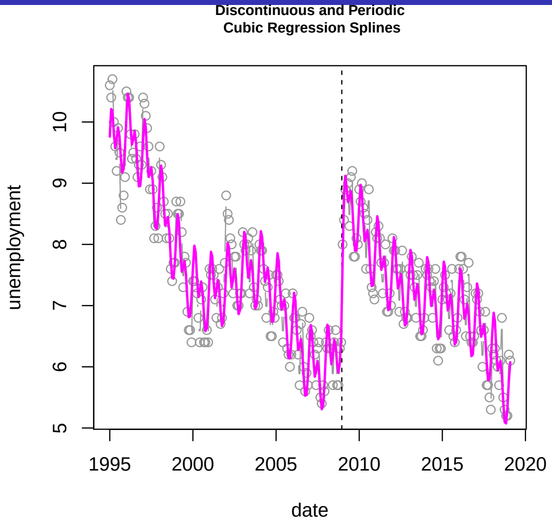
```
> per.3 <- gspline(knots=12*(1:5)/5, degree=3, smoothness=2, periodic=TRUE)
> fit3.d.p <- lm(unemployment ~ sp3.d(year) + per.3(month), data=Unemp)
```

Example: Canadian Unemployment and the 2008 Crash

Adding a Periodic Regression Spline to the Model

- Graphing the fit:

```
> plot(unemployment ~ date, data=Unemp,
+  col=gray(0.60), type="b",
+  main="Discontinuous and Periodic
+  Cubic Regression Splines", cex.main=0.75)
> abline(v=as.Date("2008-12-15"),
+        lty=2)
> lines(Unemp$date, predict(fit3.d.p), lwd=2,
+       col="magenta")
```



- Some model comparisons:

```
> AIC(fit2, fit3, fit2.d, fit3.d, fit3.d.p)
         df    AIC
fit2      8 633.55
fit3      9 657.07
fit2.d   11 525.27
fit3.d   13 469.35
fit3.d.p 17 239.01
```

Example: Canadian Unemployment and the 2008 Crash

Adding a Periodic Regression Spline to the Model

- Some model comparisons:

```
> AIC(fit2, fit3, fit2.d, fit3.d, fit3.d.p)
```

```
         df    AIC
fit2      8 633.55
fit3      9 657.07
fit2.d   11 525.27
fit3.d   13 469.35
fit3.d.p 17 239.01
```

- The AIC strongly favours the cubic model with both discontinuity and a periodic component.

Room for Improvement

- The spline bases constructed by `gspline()` can be much less numerically stable than those constructed by, e.g., `bs()` or `ns()`.

Room for Improvement

- The spline bases constructed by `gspline()` can be much less numerically stable than those constructed by, e.g., `bs()` or `ns()`.
    - To make the spline bases more stable it's necessary to make them data dependent.

Room for Improvement

- The spline bases constructed by `gspline()` can be much less numerically stable than those constructed by, e.g., `bs()` or `ns()`.
  - To make the spline bases more stable it's necessary to make them data dependent.
  - We have a plan, tested but not fully implemented, to accomplish this while still maintaining an interpretable parametrization of the regression splines.

Room for Improvement

- The spline bases constructed by `gspline()` can be much less numerically stable than those constructed by, e.g., `bs()` or `ns()`.
  - To make the spline bases more stable it's necessary to make them data dependent.
  - We have a plan, tested but not fully implemented, to accomplish this while still maintaining an interpretable parametrization of the regression splines.
- We'd also like to make the `wald()` function (which is useful for testing hypotheses about regression splines but is not illustrated in this presentation) more intelligent in how it deals with spline models whose linear predictors have terms related by marginality, e.g., of the form generated by `sp(numeric.predictor)*factor`.

References

Fox, J. (2016). *Applied Regression Analysis and Generalized Linear Models*. Sage, Thousand Oaks, CA, 3rd edition.

Harrell, Jr., F. E. (2015). *Regression Modeling Strategies, With Applications to Linear Models, Logistic Regression, and Survival Analysis*. Springer, New York, second edition.