Imperial College London

Sensible parameters for polynomials and other splines

Roger B. Newson
r.newson@imperial.ac.uk
http://www.imperial.ac.uk/nhli/r.newson/

National Heart and Lung Institute
Imperial College London

17th UK Stata Users' Group Meeting, 15–16 September, 2011

Polynomials and other splines

- A k**th-degree spline** is a function from the X-axis to the Y-axis, defined using an ascending sequence of **knots** $s_0 < s_1 < \ldots < s_q$ on the X-axis.
- (*Typically*, the sequence of knots is assumed to be part of an extended sequence of form $\ldots s_{-1} < s_0 < \ldots < s_q < s_{q+1} \ldots$ extending outwards to $\pm\infty$.)
- In each interval $s_j \leq x < s_{j+1}$ between two successive knots, the spline is equal to a kth degree polynomial.
- (*Therefore*, a polynomial, restricted to a bounded interval, is a special case of a spline, with knots at the boundaries.)
- At each knot s_j, the first $k-1$ derivatives of the spline are continuous.
- *Therefore*, a spline of degree 0 is a step function, a spline of degree 1 is linearly interpolated between the knots, and splines of degree 2, 3 and higher are interpolated as curves.

Polynomials and other splines

- A k**th-degree spline** is a function from the X-axis to the Y-axis, defined using an ascending sequence of **knots** $s_0 < s_1 < \ldots < s_q$ on the X-axis.
- (*Typically*, the sequence of knots is assumed to be part of an extended sequence of form $\ldots s_{-1} < s_0 < \ldots < s_q < s_{q+1} \ldots$ extending outwards to $\pm\infty$.)
- In each interval $s_j \le x < s_{j+1}$ between two successive knots, the spline is equal to a kth degree polynomial.
- (*Therefore*, a polynomial, restricted to a bounded interval, is a special case of a spline, with knots at the boundaries.)
- At each knot s_j, the first $k-1$ derivatives of the spline are continuous.
- *Therefore*, a spline of degree 0 is a step function, a spline of degree 1 is linearly interpolated between the knots, and splines of degree 2, 3 and higher are interpolated as curves.

Polynomials and other splines

- A k**th-degree spline** is a function from the X-axis to the Y-axis, defined using an ascending sequence of **knots** $s_0 < s_1 < \ldots < s_q$ on the X-axis.
- (*Typically*, the sequence of knots is assumed to be part of an extended sequence of form $\ldots s_{-1} < s_0 < \ldots < s_q < s_{q+1} \ldots$ extending outwards to $\pm\infty$.)
- In each interval $s_j \leq x < s_{j+1}$ between two successive knots, the spline is equal to a kth degree polynomial.
- (*Therefore*, a polynomial, restricted to a bounded interval, is a special case of a spline, with knots at the boundaries.)
- At each knot s_j, the first $k - 1$ derivatives of the spline are continuous.
- *Therefore*, a spline of degree 0 is a step function, a spline of degree 1 is linearly interpolated between the knots, and splines of degree 2, 3 and higher are interpolated as curves.

Polynomials and other splines

- A k**th-degree spline** is a function from the X-axis to the Y-axis, defined using an ascending sequence of **knots** $s_0 < s_1 < \ldots < s_q$ on the X-axis.
- (*Typically*, the sequence of knots is assumed to be part of an extended sequence of form $\ldots s_{-1} < s_0 < \ldots < s_q < s_{q+1} \ldots$ extending outwards to $\pm\infty$.)
- In each interval $s_j \leq x < s_{j+1}$ between two successive knots, the spline is equal to a kth degree polynomial.
- (*Therefore*, a polynomial, restricted to a bounded interval, is a special case of a spline, with knots at the boundaries.)
- At each knot s_j, the first $k-1$ derivatives of the spline are continuous.
- *Therefore*, a spline of degree 0 is a step function, a spline of degree 1 is linearly interpolated between the knots, and splines of degree 2, 3 and higher are interpolated as curves.

Polynomials and other splines

- A k**th-degree spline** is a function from the X-axis to the Y-axis, defined using an ascending sequence of **knots** $s_0 < s_1 < \ldots < s_q$ on the X-axis.
- (*Typically*, the sequence of knots is assumed to be part of an extended sequence of form $\ldots s_{-1} < s_0 < \ldots < s_q < s_{q+1} \ldots$ extending outwards to $\pm\infty$.)
- In each interval $s_j \leq x < s_{j+1}$ between two successive knots, the spline is equal to a kth degree polynomial.
- (*Therefore*, a polynomial, restricted to a bounded interval, is a special case of a spline, with knots at the boundaries.)
- At each knot s_j, the first $k-1$ derivatives of the spline are continuous.
- *Therefore*, a spline of degree 0 is a step function, a spline of degree 1 is linearly interpolated between the knots, and splines of degree 2, 3 and higher are interpolated as curves.

Polynomials and other splines

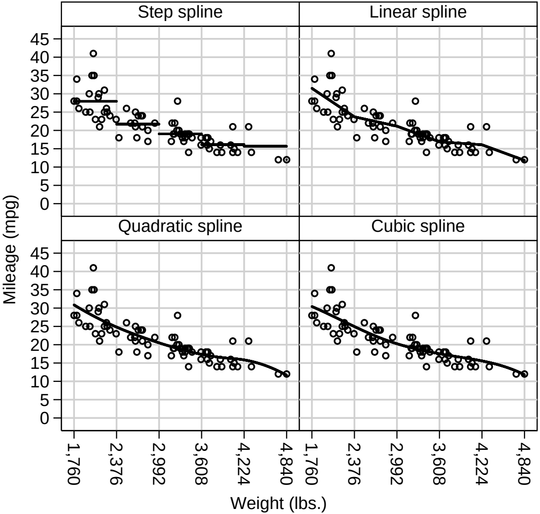
- A k**th-degree spline** is a function from the X-axis to the Y-axis, defined using an ascending sequence of **knots** $s_0 < s_1 < \ldots < s_q$ on the X-axis.
- (*Typically*, the sequence of knots is assumed to be part of an extended sequence of form $\ldots s_{-1} < s_0 < \ldots < s_q < s_{q+1} \ldots$ extending outwards to $\pm\infty$.)
- In each interval $s_j \leq x < s_{j+1}$ between two successive knots, the spline is equal to a kth degree polynomial.
- (*Therefore*, a polynomial, restricted to a bounded interval, is a special case of a spline, with knots at the boundaries.)
- At each knot s_j, the first $k-1$ derivatives of the spline are continuous.
- *Therefore*, a spline of degree 0 is a step function, a spline of degree 1 is linearly interpolated between the knots, and splines of degree 2, 3 and higher are interpolated as curves.

Polynomials and other splines

- A k**th-degree spline** is a function from the X-axis to the Y-axis, defined using an ascending sequence of **knots** $s_0 < s_1 < \ldots < s_q$ on the X-axis.
- (*Typically*, the sequence of knots is assumed to be part of an extended sequence of form $\ldots s_{-1} < s_0 < \ldots < s_q < s_{q+1} \ldots$ extending outwards to $\pm\infty$.)
- In each interval $s_j \leq x < s_{j+1}$ between two successive knots, the spline is equal to a kth degree polynomial.
- (*Therefore*, a polynomial, restricted to a bounded interval, is a special case of a spline, with knots at the boundaries.)
- At each knot s_j, the first $k - 1$ derivatives of the spline are continuous.
- *Therefore*, a spline of degree 0 is a step function, a spline of degree 1 is linearly interpolated between the knots, and splines of degree 2, 3 and higher are interpolated as curves.

Spline models for `mpg` with respect to `weight` in the `auto` data

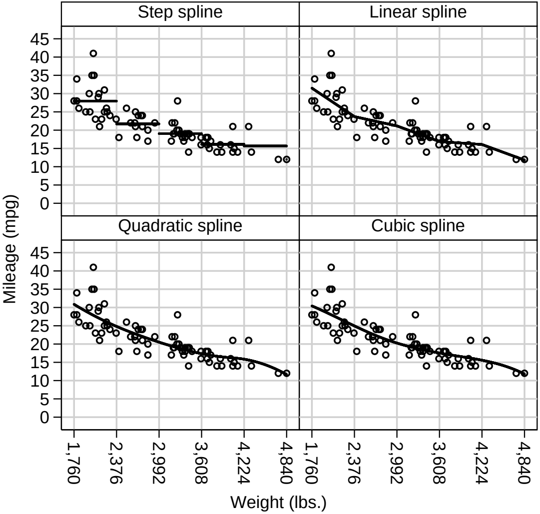
- The plots illustrate splines of degree 0, 1, 2 and 3.
- The degree–zero spline is a step function, constant within intervals.
- The degree–1 spline is interpolated linearly between reference points on the *X*–axis.
- The degree–2 and degree–3 splines are interpolated quadratically and cubically.



Graphs by Degree of spline

Spline models for `mpg` with respect to `weight` in the `auto` data

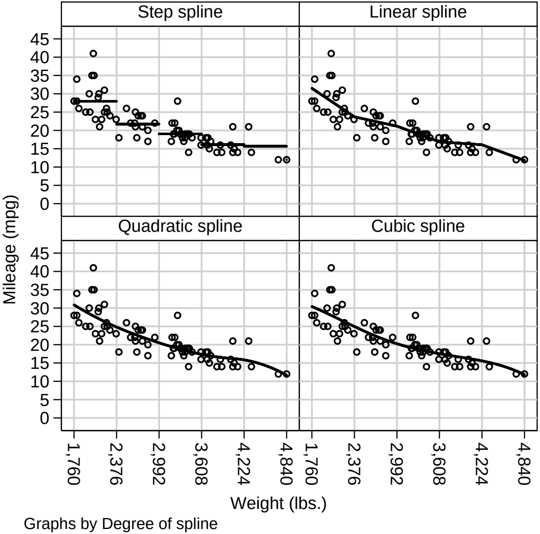
- The plots illustrate splines of degree 0, 1, 2 and 3.
- The degree–zero spline is a step function, constant within intervals.
- The degree–1 spline is interpolated linearly between reference points on the *X*–axis.
- The degree–2 and degree–3 splines are interpolated quadratically and cubically.



Graphs by Degree of spline

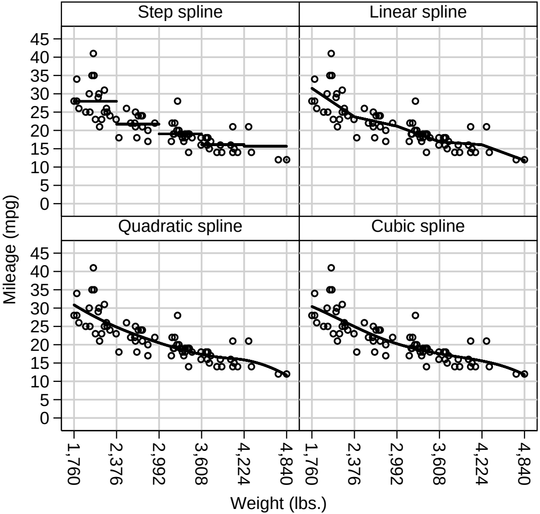
Spline models for `mpg` with respect to `weight` in the `auto` data

- The plots illustrate splines of degree 0, 1, 2 and 3.
- The degree–zero spline is a step function, constant within intervals.
- The degree–1 spline is interpolated linearly between reference points on the *X*–axis.
- The degree–2 and degree–3 splines are interpolated quadratically and cubically.



Spline models for `mpg` with respect to `weight` in the `auto` data

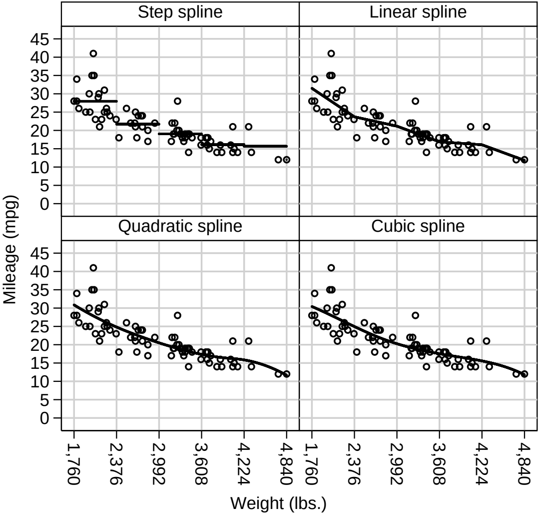
- The plots illustrate splines of degree 0, 1, 2 and 3.
- The degree–zero spline is a step function, constant within intervals.
- The degree–1 spline is interpolated linearly between reference points on the *X*–axis.
- The degree–2 and degree–3 splines are interpolated quadratically and cubically.



Graphs by Degree of spline

Spline models for `mpg` with respect to `weight` in the `auto` data

- The plots illustrate splines of degree 0, 1, 2 and 3.
- The degree–zero spline is a step function, constant within intervals.
- The degree–1 spline is interpolated linearly between reference points on the *X*–axis.
- The degree–2 and degree–3 splines are interpolated quadratically and cubically.



Graphs by Degree of spline

Sensible parameters for splines

- A polynomial model, defined by $Y = \sum_{j=0}^{K} b_j X^j$, has semi–sensible parameters b_j, expressed in units such as Y–units per squared X–unit.
- Such parameters are not always easy to explain to non–mathematical colleagues.
- Other spline models are frequently parameterized in even less intuitive ways, sometimes expressed in Y–axis units per inverse X–axis unit.
- Such parameters are frequently not easy to understand, even for mathematicians, who sometimes use nonsensical jargon such as "non–parametric regression" to describe these models.
- It would be easier if the parameters were values of the spline at reference points on the X–axis.
- Or, alternatively, differences or ratios between spline values at reference points and spline values at a base reference point.

Sensible parameters for splines

- A polynomial model, defined by $Y = \sum_{j=0}^{K} b_j X^j$, has semi–sensible parameters b_j, expressed in units such as Y–units per squared X–unit.
- Such parameters are not always easy to explain to non–mathematical colleagues.
- Other spline models are frequently parameterized in even less intuitive ways, sometimes expressed in Y–axis units per inverse X–axis unit.
- Such parameters are frequently not easy to understand, even for mathematicians, who sometimes use nonsensical jargon such as “non–parametric regression” to describe these models.
- It would be easier if the parameters were values of the spline at reference points on the X–axis.
- Or, alternatively, differences or ratios between spline values at reference points and spline values at a base reference point.

Sensible parameters for splines

- A polynomial model, defined by $Y = \sum_{j=0}^{K} b_j X^j$, has semi–sensible parameters b_j, expressed in units such as Y–units per squared X–unit.
- Such parameters are not always easy to explain to non–mathematical colleagues.
- Other spline models are frequently parameterized in even less intuitive ways, sometimes expressed in Y–axis units per inverse X–axis unit.
- Such parameters are frequently not easy to understand, even for mathematicians, who sometimes use nonsensical jargon such as "non–parametric regression" to describe these models.
- It would be easier if the parameters were values of the spline at reference points on the X–axis.
- Or, alternatively, differences or ratios between spline values at reference points and spline values at a base reference point.

Sensible parameters for splines

- A polynomial model, defined by $Y = \sum_{j=0}^{K} b_j X^j$, has semi–sensible parameters b_j, expressed in units such as Y–units per squared X–unit.
- Such parameters are not always easy to explain to non–mathematical colleagues.
- Other spline models are frequently parameterized in even less intuitive ways, sometimes expressed in Y–axis units per inverse X–axis unit.
- Such parameters are frequently not easy to understand, even for mathematicians, who sometimes use nonsensical jargon such as "non–parametric regression" to describe these models.
- It would be easier if the parameters were values of the spline at reference points on the X–axis.
- Or, alternatively, differences or ratios between spline values at reference points and spline values at a base reference point.

Sensible parameters for splines

- A polynomial model, defined by $Y = \sum_{j=0}^{K} b_j X^j$, has semi–sensible parameters b_j, expressed in units such as Y–units per squared X–unit.
- Such parameters are not always easy to explain to non–mathematical colleagues.
- Other spline models are frequently parameterized in even less intuitive ways, sometimes expressed in Y–axis units per inverse X–axis unit.
- Such parameters are frequently not easy to understand, even for mathematicians, who sometimes use nonsensical jargon such as "non–parametric regression" to describe these models.
- It would be easier if the parameters were values of the spline at reference points on the X–axis.
- Or, alternatively, differences or ratios between spline values at reference points and spline values at a base reference point.

Sensible parameters for splines

- A polynomial model, defined by $Y = \sum_{j=0}^{K} b_j X^j$, has semi–sensible parameters b_j, expressed in units such as Y–units per squared X–unit.
- Such parameters are not always easy to explain to non–mathematical colleagues.
- Other spline models are frequently parameterized in even less intuitive ways, sometimes expressed in Y–axis units per inverse X–axis unit.
- Such parameters are frequently not easy to understand, even for mathematicians, who sometimes use nonsensical jargon such as “non–parametric regression” to describe these models.
- It would be easier if the parameters were values of the spline at reference points on the X–axis.
- Or, alternatively, differences or ratios between spline values at reference points and spline values at a base reference point.

Sensible parameters for splines

- A polynomial model, defined by $Y = \sum_{j=0}^{K} b_j X^j$, has semi–sensible parameters b_j, expressed in units such as Y–units per squared X–unit.
- Such parameters are not always easy to explain to non–mathematical colleagues.
- Other spline models are frequently parameterized in even less intuitive ways, sometimes expressed in Y–axis units per inverse X–axis unit.
- Such parameters are frequently not easy to understand, even for mathematicians, who sometimes use nonsensical jargon such as "non–parametric regression" to describe these models.
- It would be easier if the parameters were values of the spline at reference points on the X–axis.
- Or, alternatively, differences or ratios between spline values at reference points and spline values at a base reference point.

The `flexcurv` module of the `bspline` package

- The SSC package `bspline`, introduced in STB[2] and at the 2001 UK Stata Users' Meeting[3], now has 3 modules.
- The module `bspline` inputs an X–variable and a sequence of knots, and generates a basis of Schoenberg B–splines[4].
- The module `frencurv` uses `bspline`, followed by matrix inversion, to generate a basis of reference splines, whose corresponding regression parameters are spline values at reference points on the X–axis.
- `frencurv` chooses default knots equal to reference points for odd–degree splines, and to midpoints between reference points for even–degree splines.
- For splines of degree $k > 1$, this implies reference points off the edge of the X–axis, which are *not* easy to explain.
- The recently–added module `flexcurv` uses `frencurv` to generate reference splines for within–range reference points, using automatically–generated regularly–spaced knots.

The `flexcurv` module of the `bspline` package

- The SSC package `bspline`, introduced in STB[2] and at the 2001 UK Stata Users' Meeting[3], now has 3 modules.
- The module `bspline` inputs an X–variable and a sequence of knots, and generates a basis of Schoenberg B–splines[4].
- The module `frencurv` uses `bspline`, followed by matrix inversion, to generate a basis of reference splines, whose corresponding regression parameters are spline values at reference points on the X–axis.
- `frencurv` chooses default knots equal to reference points for odd–degree splines, and to midpoints between reference points for even–degree splines.
- For splines of degree $k > 1$, this implies reference points off the edge of the X–axis, which are *not* easy to explain.
- The recently–added module `flexcurv` uses `frencurv` to generate reference splines for within–range reference points, using automatically–generated regularly–spaced knots.

The **flexcurv** module of the **bspline** package

- The SSC package `bspline`, introduced in STB[2] and at the 2001 UK Stata Users' Meeting[3], now has 3 modules.
- The module `bspline` inputs an *X*–variable and a sequence of knots, and generates a basis of Schoenberg *B*–splines[4].
- The module `frencurv` uses `bspline`, followed by matrix inversion, to generate a basis of reference splines, whose corresponding regression parameters are spline values at reference points on the *X*–axis.
- `frencurv` chooses default knots equal to reference points for odd–degree splines, and to midpoints between reference points for even–degree splines.
- For splines of degree $k > 1$, this implies reference points off the edge of the *X*–axis, which are *not* easy to explain.
- The recently–added module `flexcurv` uses `frencurv` to generate reference splines for within–range reference points, using automatically–generated regularly–spaced knots.

The `flexcurv` module of the `bspline` package

- The SSC package `bspline`, introduced in STB[2] and at the 2001 UK Stata Users' Meeting[3], now has 3 modules.
- The module `bspline` inputs an *X*–variable and a sequence of knots, and generates a basis of Schoenberg *B*–splines[4].
- The module `frencurv` uses `bspline`, followed by matrix inversion, to generate a basis of reference splines, whose corresponding regression parameters are spline values at reference points on the *X*–axis.
- `frencurv` chooses default knots equal to reference points for odd–degree splines, and to midpoints between reference points for even–degree splines.
- For splines of degree $k > 1$, this implies reference points off the edge of the *X*–axis, which are *not* easy to explain.
- The recently–added module `flexcurv` uses `frencurv` to generate reference splines for within–range reference points, using automatically–generated regularly–spaced knots.

The `flexcurv` module of the `bspline` package

- The SSC package `bspline`, introduced in STB[2] and at the 2001 UK Stata Users' Meeting[3], now has 3 modules.
- The module `bspline` inputs an *X*–variable and a sequence of knots, and generates a basis of Schoenberg *B*–splines[4].
- The module `frencurv` uses `bspline`, followed by matrix inversion, to generate a basis of reference splines, whose corresponding regression parameters are spline values at reference points on the *X*–axis.
- `frencurv` chooses default knots equal to reference points for odd–degree splines, and to midpoints between reference points for even–degree splines.
- For splines of degree $k > 1$, this implies reference points off the edge of the *X*–axis, which are *not* easy to explain.
- The recently–added module `flexcurv` uses `frencurv` to generate reference splines for within–range reference points, using automatically–generated regularly–spaced knots.

The `flexcurv` module of the `bspline` package

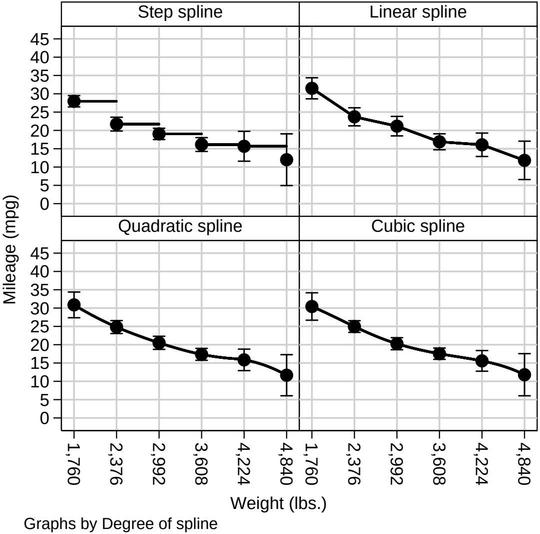
- The SSC package `bspline`, introduced in STB[2] and at the 2001 UK Stata Users' Meeting[3], now has 3 modules.
- The module `bspline` inputs an *X*–variable and a sequence of knots, and generates a basis of Schoenberg *B*–splines[4].
- The module `frencurv` uses `bspline`, followed by matrix inversion, to generate a basis of reference splines, whose corresponding regression parameters are spline values at reference points on the *X*–axis.
- `frencurv` chooses default knots equal to reference points for odd–degree splines, and to midpoints between reference points for even–degree splines.
- For splines of degree $k > 1$, this implies reference points off the edge of the *X*–axis, which are *not* easy to explain.
- The recently–added module `flexcurv` uses `frencurv` to generate reference splines for within–range reference points, using automatically–generated regularly–spaced knots.

The `flexcurv` module of the `bspline` package

- The SSC package `bspline`, introduced in STB[2] and at the 2001 UK Stata Users' Meeting[3], now has 3 modules.
- The module `bspline` inputs an *X*–variable and a sequence of knots, and generates a basis of Schoenberg *B*–splines[4].
- The module `frencurv` uses `bspline`, followed by matrix inversion, to generate a basis of reference splines, whose corresponding regression parameters are spline values at reference points on the *X*–axis.
- `frencurv` chooses default knots equal to reference points for odd–degree splines, and to midpoints between reference points for even–degree splines.
- For splines of degree $k > 1$, this implies reference points off the edge of the *X*–axis, which are *not* easy to explain.
- The recently–added module `flexcurv` uses `frencurv` to generate reference splines for within–range reference points, using automatically–generated regularly–spaced knots.

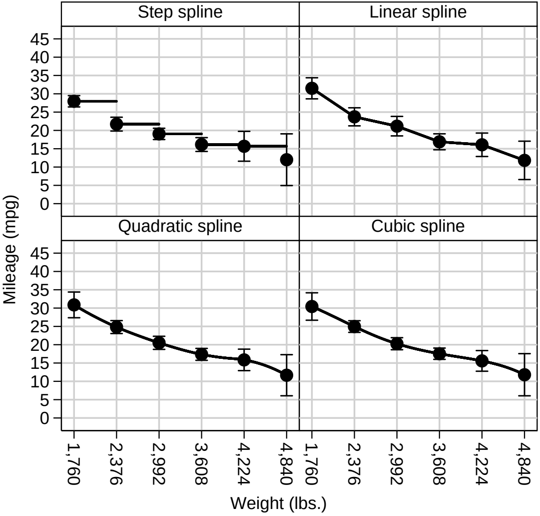
Parameters for splines for `mpg` with respect to `weight` in the `auto` data

- The plots again illustrate splines of degree 0, 1, 2 and 3.
- *However*, this time, instead of the observed values, we see 95% confidence intervals for the spline parameters.
- These parameters are values of the splines at the reference points indicated on the X–axis.



Parameters for splines for `mpg` with respect to `weight` in the `auto` data

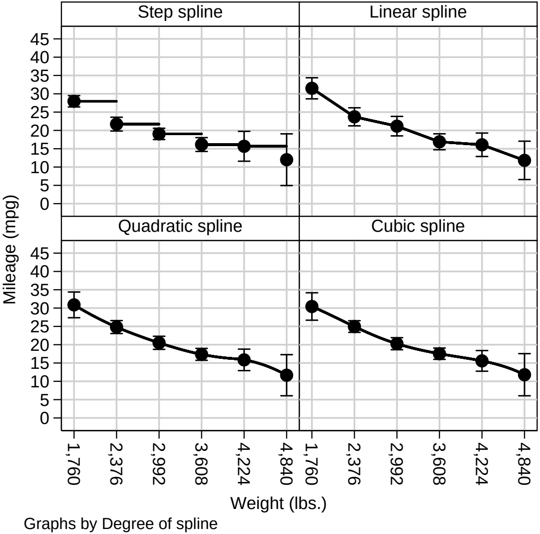
- The plots again illustrate splines of degree 0, 1, 2 and 3.
- *However*, this time, instead of the observed values, we see 95% confidence intervals for the spline parameters.
- These parameters are values of the splines at the reference points indicated on the *X*–axis.



Graphs by Degree of spline

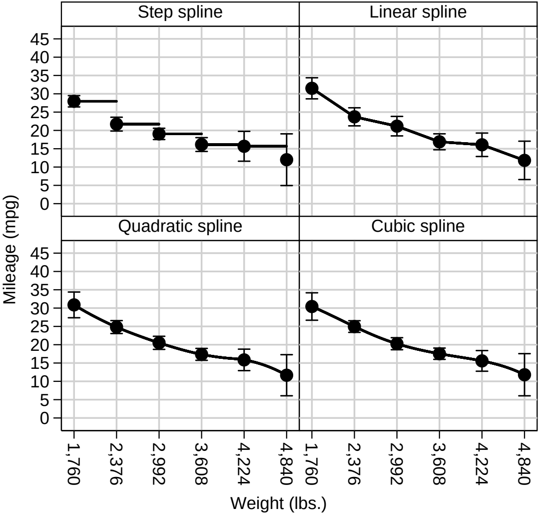
Parameters for splines for `mpg` with respect to `weight` in the `auto` data

- The plots again illustrate splines of degree 0, 1, 2 and 3.
- *However*, this time, instead of the observed values, we see 95% confidence intervals for the spline parameters.
- These parameters are values of the splines at the reference points indicated on the *X*–axis.



Parameters for splines for `mpg` with respect to `weight` in the `auto` data

- The plots again illustrate splines of degree 0, 1, 2 and 3.
- *However*, this time, instead of the observed values, we see 95% confidence intervals for the spline parameters.
- These parameters are values of the splines at the reference points indicated on the *X*–axis.



Graphs by Degree of spline

So how do we use `flexcurv`?

- `flexcurv` inputs an *X*–variable, a list of reference points on the *X*–axis, and a user–specified spline degree (or power).
- It outputs a **basis** of new variables, one for each reference point, known as **reference splines**.
- (It also calculates a sequence of regularly–spaced knots, which the user need not think about.)
- Splines of the specified degree, with the determined sequence of knots, are linear combinations of these reference splines.
- *Therefore*, the reference splines can be included in a design matrix, for input to an estimation command, using the `noconst` option.
- *So*, reference splines are an extension, to continuous factors, of the indicator (or "dummy") variables generated for discrete factors, using `xi:` or `tabulate`.

So how do we use `flexcurv`?

- `flexcurv` inputs an *X*–variable, a list of reference points on the *X*–axis, and a user–specified spline degree (or power).
- It outputs a **basis** of new variables, one for each reference point, known as **reference splines**.
- (It also calculates a sequence of regularly–spaced knots, which the user need not think about.)
- Splines of the specified degree, with the determined sequence of knots, are linear combinations of these reference splines.
- *Therefore*, the reference splines can be included in a design matrix, for input to an estimation command, using the `noconst` option.
- *So*, reference splines are an extension, to continuous factors, of the indicator (or "dummy") variables generated for discrete factors, using `xi:` or `tabulate`.

So how do we use `flexcurv`?

- `flexcurv` inputs an *X*–variable, a list of reference points on the *X*–axis, and a user–specified spline degree (or power).
- It outputs a **basis** of new variables, one for each reference point, known as **reference splines**.
- (It also calculates a sequence of regularly–spaced knots, which the user need not think about.)
- Splines of the specified degree, with the determined sequence of knots, are linear combinations of these reference splines.
- *Therefore*, the reference splines can be included in a design matrix, for input to an estimation command, using the `noconst` option.
- *So*, reference splines are an extension, to continuous factors, of the indicator (or "dummy") variables generated for discrete factors, using `xi:` or `tabulate`.

So how do we use `flexcurv`?

- `flexcurv` inputs an *X*–variable, a list of reference points on the *X*–axis, and a user–specified spline degree (or power).
- It outputs a **basis** of new variables, one for each reference point, known as **reference splines**.
- (It also calculates a sequence of regularly–spaced knots, which the user need not think about.)
- Splines of the specified degree, with the determined sequence of knots, are linear combinations of these reference splines.
- *Therefore*, the reference splines can be included in a design matrix, for input to an estimation command, using the `noconst` option.
- *So*, reference splines are an extension, to continuous factors, of the indicator (or "dummy") variables generated for discrete factors, using `xi:` or `tabulate`.

So how do we use `flexcurv`?

- `flexcurv` inputs an *X*–variable, a list of reference points on the *X*–axis, and a user–specified spline degree (or power).
- It outputs a **basis** of new variables, one for each reference point, known as **reference splines**.
- (It also calculates a sequence of regularly–spaced knots, which the user need not think about.)
- Splines of the specified degree, with the determined sequence of knots, are linear combinations of these reference splines.
- *Therefore*, the reference splines can be included in a design matrix, for input to an estimation command, using the `noconst` option.
- *So*, reference splines are an extension, to continuous factors, of the indicator (or "dummy") variables generated for discrete factors, using `xi:` or `tabulate`.

So how do we use `flexcurv`?

- `flexcurv` inputs an *X*–variable, a list of reference points on the *X*–axis, and a user–specified spline degree (or power).
- It outputs a **basis** of new variables, one for each reference point, known as **reference splines**.
- (It also calculates a sequence of regularly–spaced knots, which the user need not think about.)
- Splines of the specified degree, with the determined sequence of knots, are linear combinations of these reference splines.
- *Therefore*, the reference splines can be included in a design matrix, for input to an estimation command, using the `noconst` option.
- *So*, reference splines are an extension, to continuous factors, of the indicator (or "dummy") variables generated for discrete factors, using `xi:` or `tabulate`.

So how do we use `flexcurv`?

- `flexcurv` inputs an *X*–variable, a list of reference points on the *X*–axis, and a user–specified spline degree (or power).
- It outputs a **basis** of new variables, one for each reference point, known as **reference splines**.
- (It also calculates a sequence of regularly–spaced knots, which the user need not think about.)
- Splines of the specified degree, with the determined sequence of knots, are linear combinations of these reference splines.
- *Therefore*, the reference splines can be included in a design matrix, for input to an estimation command, using the `noconst` option.
- *So*, reference splines are an extension, to continuous factors, of the indicator (or "dummy") variables generated for discrete factors, using `xi:` or `tabulate`.

Example: Cubic spline of `mpg` with respect to `weight`

We begin by loading the `auto` data, and use `flexcurv` to input a list of 6 reference points and generate a basis of 6 cubic reference splines in `weight`, which we then `describe`:

```
. flexcurv, xvar(weight) refpts(1760(616)4840) power(3)
>   generate(sp_);

. describe sp_*;

              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
sp_1            float  %8.4f                  Spline at 1,760
sp_2            float  %8.4f                  Spline at 2,376
sp_3            float  %8.4f                  Spline at 2,992
sp_4            float  %8.4f                  Spline at 3,608
sp_5            float  %8.4f                  Spline at 4,224
sp_6            float  %8.4f                  Spline at 4,840
```

Note that each reference spline has a variable label, indicating its reference point on the `weight` axis.

Regression of **`mpg`** with respect to the splines in **`weight`**

We then use `regress`, with the `noconst` option, to fit a linear regression model of `mpg` with respect to the 6 reference splines:

```
. regress mpg sp_*, noconst nohead;
------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        sp_1 |   30.42892   1.875043    16.23   0.000     26.68733    34.17051
        sp_2 |   24.95794   .7959588    31.36   0.000     23.36963    26.54625
        sp_3 |   20.26864   .8231831    24.62   0.000       18.626    21.91127
        sp_4 |   17.54179   .7730685    22.69   0.000     15.99916    19.08443
        sp_5 |   15.57965   1.413921    11.02   0.000     12.75821    18.40108
        sp_6 |   11.80283   2.882191     4.10   0.000     6.051511    17.55416
------------------------------------------------------------------------------
```

The estimated parameter for each spline `sp_1` to `sp_6` is the conditional mean of `mpg` (in miles per gallon) under the spline model, assuming that `weight` is equal to the corresponding reference point. *However…*

Listing of the conditional means using **parmest**

...we can present these estimates and confidence limits more informatively using the SSC package `parmest`, as follows:

```
. parmest, label list(parm label estimate min* max*, sepa(0))
>   format(estimate min* max* %8.2f);

     +------------------------------------------------------+
     | parm              label   estimate   min95   max95 |
     |------------------------------------------------------|
  1. | sp_1   Spline at 1,760      30.43   26.69   34.17 |
  2. | sp_2   Spline at 2,376      24.96   23.37   26.55 |
  3. | sp_3   Spline at 2,992      20.27   18.63   21.91 |
  4. | sp_4   Spline at 3,608      17.54   16.00   19.08 |
  5. | sp_5   Spline at 4,224      15.58   12.76   18.40 |
  6. | sp_6   Spline at 4,840      11.80    6.05   17.55 |
     +------------------------------------------------------+
```

The `label` and `list()` options allow us to see, instantly, which conditional mean (expressed in miles per gallon) belongs to each value of `weight` (expressed in US or Imperial pounds). And the `format()` option formats these parameters sensibly.

Useful features of the reference splines in a basis

- Any spline of the specified degree, with the calculated regularly–spaced knots, is equal to a linear combination of the reference splines in the basis.
- (So, predicted values of the spline at non–reference points are equal to linear combinations of the values of the spline at the reference points.)
- In particular, the unit vector is a spline, whose co–ordinates in the reference splines are all 1.
- (In other words, the reference splines sum to 1.)
- And each reference spline is equal to 1 at its own reference point, and to 0 at all other reference points.

Useful features of the reference splines in a basis

- Any spline of the specified degree, with the calculated regularly–spaced knots, is equal to a linear combination of the reference splines in the basis.
- (So, predicted values of the spline at non–reference points are equal to linear combinations of the values of the spline at the reference points.)
- In particular, the unit vector is a spline, whose co–ordinates in the reference splines are all 1.
- (In other words, the reference splines sum to 1.)
- And each reference spline is equal to 1 at its own reference point, and to 0 at all other reference points.

Useful features of the reference splines in a basis

- Any spline of the specified degree, with the calculated regularly–spaced knots, is equal to a linear combination of the reference splines in the basis.
- (So, predicted values of the spline at non–reference points are equal to linear combinations of the values of the spline at the reference points.)
- In particular, the unit vector is a spline, whose co–ordinates in the reference splines are all 1.
- (In other words, the reference splines sum to 1.)
- And each reference spline is equal to 1 at its own reference point, and to 0 at all other reference points.

Useful features of the reference splines in a basis

- Any spline of the specified degree, with the calculated regularly–spaced knots, is equal to a linear combination of the reference splines in the basis.
- (So, predicted values of the spline at non–reference points are equal to linear combinations of the values of the spline at the reference points.)
- In particular, the unit vector is a spline, whose co–ordinates in the reference splines are all 1.
- (In other words, the reference splines sum to 1.)
- And each reference spline is equal to 1 at its own reference point, and to 0 at all other reference points.

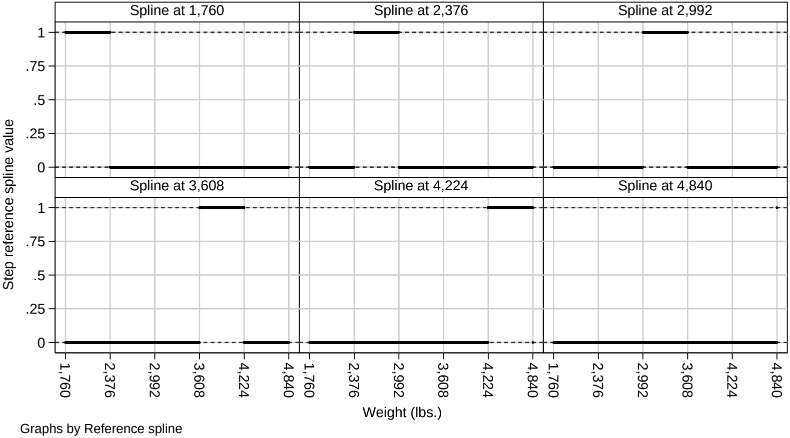
Useful features of the reference splines in a basis

- Any spline of the specified degree, with the calculated regularly–spaced knots, is equal to a linear combination of the reference splines in the basis.
- (So, predicted values of the spline at non–reference points are equal to linear combinations of the values of the spline at the reference points.)
- In particular, the unit vector is a spline, whose co–ordinates in the reference splines are all 1.
- (In other words, the reference splines sum to 1.)
- And each reference spline is equal to 1 at its own reference point, and to 0 at all other reference points.

Useful features of the reference splines in a basis

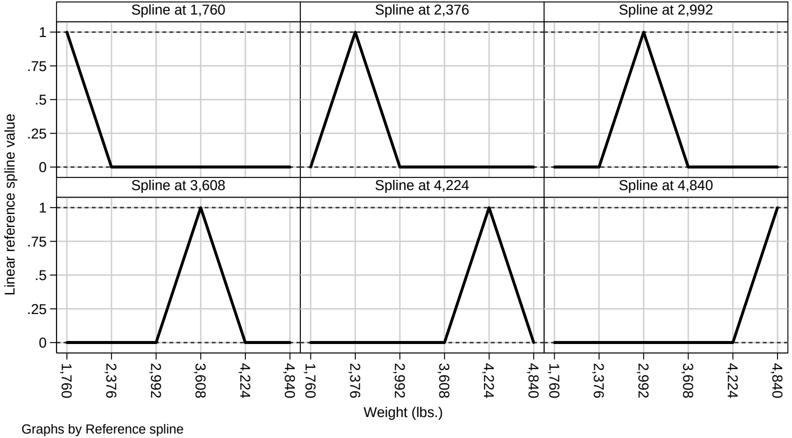
- Any spline of the specified degree, with the calculated regularly–spaced knots, is equal to a linear combination of the reference splines in the basis.
- (So, predicted values of the spline at non–reference points are equal to linear combinations of the values of the spline at the reference points.)
- In particular, the unit vector is a spline, whose co–ordinates in the reference splines are all 1.
- (In other words, the reference splines sum to 1.)
- And each reference spline is equal to 1 at its own reference point, and to 0 at all other reference points.

Reference splines of degree 0 with respect to `weight`



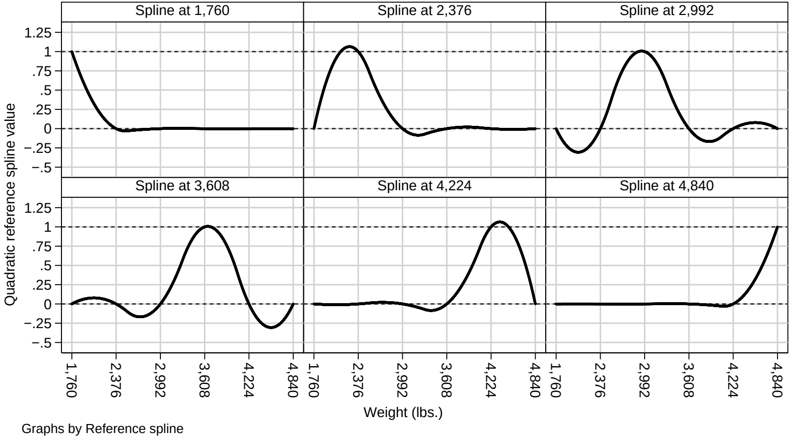
These splines are set membership functions (or dummies) for half–open intervals, beginning at a reference point (labelled on the *X*–axis), and ending "just before" the next reference point.

Reference splines of degree 1 with respect to `weight`



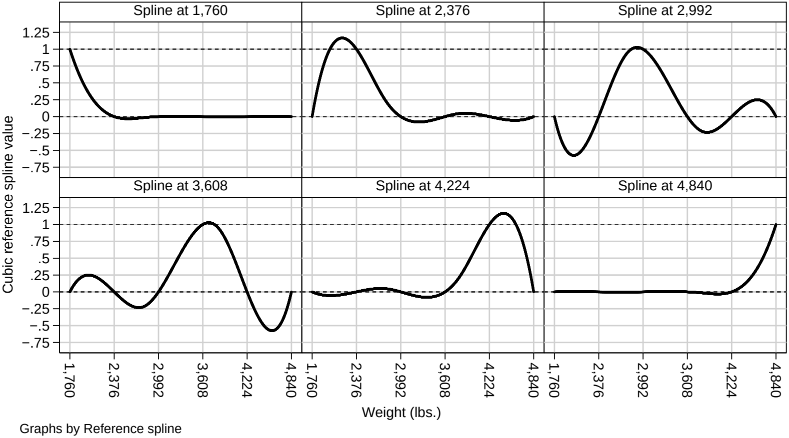
These splines are fuzzy–set membership functions[1], indicating membership (on a scale from 0 to 1) of fuzzy intervals, each centered at a reference point, and extending (in a fuzzy way) from the previous reference point to the next reference point.

Reference splines of degree 2 with respect to `weight`



These splines are curved and complicated, and less localized than the previous reference splines. *However*, each spline is still 1 at its own reference point, and 0 at all other reference points.

Reference splines of degree 3 with respect to `weight`



These splines are even more curved and complicated, and even less localized. *However*, we still see that each spline is 1 at its own reference point, and 0 at all other reference points.

So what is useful about these features?

- These features imply that, if we start with a reference–spline basis, exclude the reference spline for a base reference point, and include the unit vector, then the resulting set of splines is also a basis of the same spline space.
- The parameter corresponding to the unit vector is equal to the value of the spline at the base reference point.
- And the parameter corresponding to any other reference spline is the difference between the value of the spline at its reference point and the value of the spline at the base reference point.
- `flexcurv` and `frencurv` have an option `omit(#)` for Stata Version 10 users, causing a base reference spline to be dropped.
- They also have an option `base(#)` for Stata Version 11 or 12 users, causing the base reference spline to be set to zero.
- Either way, the generated splines can be input to an estimation command, *without* the `noconst` option.

So what is useful about these features?

- These features imply that, if we start with a reference–spline basis, exclude the reference spline for a base reference point, and include the unit vector, then the resulting set of splines is also a basis of the same spline space.
- The parameter corresponding to the unit vector is equal to the value of the spline at the base reference point.
- And the parameter corresponding to any other reference spline is the difference between the value of the spline at its reference point and the value of the spline at the base reference point.
- `flexcurv` and `frencurv` have an option `omit(#)` for Stata Version 10 users, causing a base reference spline to be dropped.
- They also have an option `base(#)` for Stata Version 11 or 12 users, causing the base reference spline to be set to zero.
- Either way, the generated splines can be input to an estimation command, *without* the `noconst` option.

So what is useful about these features?

- These features imply that, if we start with a reference–spline basis, exclude the reference spline for a base reference point, and include the unit vector, then the resulting set of splines is also a basis of the same spline space.
- The parameter corresponding to the unit vector is equal to the value of the spline at the base reference point.
- And the parameter corresponding to any other reference spline is the difference between the value of the spline at its reference point and the value of the spline at the base reference point.
- `flexcurv` and `frencurv` have an option `omit(#)` for Stata Version 10 users, causing a base reference spline to be dropped.
- They also have an option `base(#)` for Stata Version 11 or 12 users, causing the base reference spline to be set to zero.
- Either way, the generated splines can be input to an estimation command, *without* the `noconst` option.

So what is useful about these features?

- These features imply that, if we start with a reference–spline basis, exclude the reference spline for a base reference point, and include the unit vector, then the resulting set of splines is also a basis of the same spline space.
- The parameter corresponding to the unit vector is equal to the value of the spline at the base reference point.
- And the parameter corresponding to any other reference spline is the difference between the value of the spline at its reference point and the value of the spline at the base reference point.
- `flexcurv` and `frencurv` have an option `omit(#)` for Stata Version 10 users, causing a base reference spline to be dropped.
- They also have an option `base(#)` for Stata Version 11 or 12 users, causing the base reference spline to be set to zero.
- Either way, the generated splines can be input to an estimation command, *without* the `noconst` option.

So what is useful about these features?

- These features imply that, if we start with a reference–spline basis, exclude the reference spline for a base reference point, and include the unit vector, then the resulting set of splines is also a basis of the same spline space.
- The parameter corresponding to the unit vector is equal to the value of the spline at the base reference point.
- And the parameter corresponding to any other reference spline is the difference between the value of the spline at its reference point and the value of the spline at the base reference point.
- `flexcurv` and `frencurv` have an option `omit(#)` for Stata Version 10 users, causing a base reference spline to be dropped.
- They also have an option `base(#)` for Stata Version 11 or 12 users, causing the base reference spline to be set to zero.
- Either way, the generated splines can be input to an estimation command, *without* the `noconst` option.

So what is useful about these features?

- These features imply that, if we start with a reference–spline basis, exclude the reference spline for a base reference point, and include the unit vector, then the resulting set of splines is also a basis of the same spline space.
- The parameter corresponding to the unit vector is equal to the value of the spline at the base reference point.
- And the parameter corresponding to any other reference spline is the difference between the value of the spline at its reference point and the value of the spline at the base reference point.
- `flexcurv` and `frencurv` have an option `omit(#)` for Stata Version 10 users, causing a base reference spline to be dropped.
- They also have an option `base(#)` for Stata Version 11 or 12 users, causing the base reference spline to be set to zero.
- Either way, the generated splines can be input to an estimation command, *without* the `noconst` option.

So what is useful about these features?

- These features imply that, if we start with a reference–spline basis, exclude the reference spline for a base reference point, and include the unit vector, then the resulting set of splines is also a basis of the same spline space.
- The parameter corresponding to the unit vector is equal to the value of the spline at the base reference point.
- And the parameter corresponding to any other reference spline is the difference between the value of the spline at its reference point and the value of the spline at the base reference point.
- `flexcurv` and `frencurv` have an option `omit(#)` for Stata Version 10 users, causing a base reference spline to be dropped.
- They also have an option `base(#)` for Stata Version 11 or 12 users, causing the base reference spline to be set to zero.
- Either way, the generated splines can be input to an estimation command, *without* the `noconst` option.

Example: Cubic spline of mpg with respect to weight (again)

We will now fit the model fitted before, with a revised parameterization. We use `flexcurv`, with the option `base(1760)`, to input the list of 6 reference points and generate another basis of 6 cubic reference splines, which we then `describe`:

```
. flexcurv, xvar(weight) refpts(1760(616)4840) power(3)
>   generate(esp_) base(1760);

. describe esp_*;

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------------
esp_1           byte   %8.4f                  Spline at 1,760
esp_2           float  %8.4f                  Spline at 2,376
esp_3           float  %8.4f                  Spline at 2,992
esp_4           float  %8.4f                  Spline at 3,608
esp_5           float  %8.4f                  Spline at 4,224
esp_6           float  %8.4f                  Spline at 4,840
```

Note that the reference spline `esp_1`, corresponding to the base reference point 1,760, has storage type `byte`. This is because it has been set to zero and `compress`ed.

Regression of mpg with respect to the revised basis

We then use regress, this time *without* the noconst option, to fit a linear regression model of mpg with respect to the revised reference splines:

```
. regress mpg esp_*, nohead;
note: esp_1 omitted because of collinearity
------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       esp_1 |          0  (omitted)
       esp_2 |  -5.470931   2.302385    -2.38   0.020    -10.06527   -.8765925
       esp_3 |  -10.16026   1.886437    -5.39   0.000    -13.92459   -6.395937
       esp_4 |  -12.88711    2.10172    -6.13   0.000    -17.08103   -8.693193
       esp_5 |  -14.84924   2.272157    -6.54   0.000    -19.38325   -10.31522
       esp_6 |  -18.62611   3.486792    -5.34   0.000    -25.58389   -11.66832
       _cons |   30.42891   1.875042    16.23   0.000     26.68733    34.1705
------------------------------------------------------------------------------
```

This time, the parameter _cons is the value of the spline at the base reference weight of 1,760, the parameter esp_1 is omitted because it corresponds to the base reference weight, and the other spline parameters are effects on mileage of other reference weights.

Listing of the base value and weight effects using `parmest`

We then use `parmest` again, this time with the option `omit`, to list the parameters informatively, this time with confidence limits *and* *P*–values:

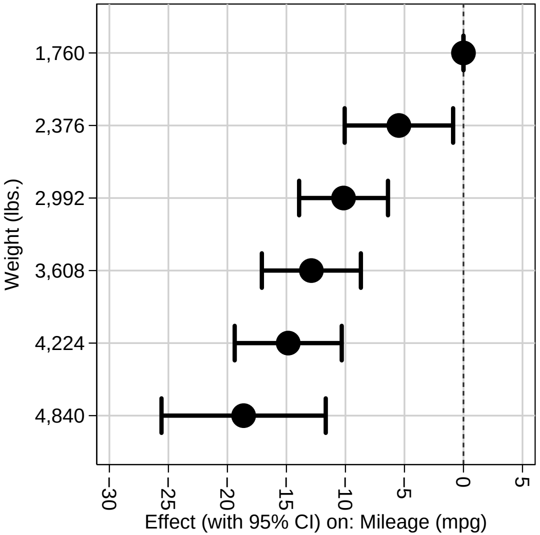
```
. parmest, label omit list(parm label omit estimate min* max* p, sepa(0))
>   format(estimate min* max* %8.2f p %-8.2g);

     +--------------------------------------------------------------------------+
     |    parm              label   omit   estimate     min95     max95   p         |
     |--------------------------------------------------------------------------|
  1. | o.esp_1   Spline at 1,760      1       0.00      0.00      0.00   .         |
  2. |   esp_2   Spline at 2,376      0      -5.47    -10.07     -0.88   .02       |
  3. |   esp_3   Spline at 2,992      0     -10.16    -13.92     -6.40   9.7e-07 |
  4. |   esp_4   Spline at 3,608      0     -12.89    -17.08     -8.69   5.0e-08 |
  5. |   esp_5   Spline at 4,224      0     -14.85    -19.38    -10.32   9.6e-09 |
  6. |   esp_6   Spline at 4,840      0     -18.63    -25.58    -11.67   1.2e-06 |
  7. |   _cons          Constant      0      30.43     26.69     34.17   4.4e-25 |
     +--------------------------------------------------------------------------+
```

We see that the parameter `_cons` is the constant term, the omitted effect of the base weight 1,760 is 0 with both confidence limits 0, and the effects of the other weights on mileage are clearly significantly negative.

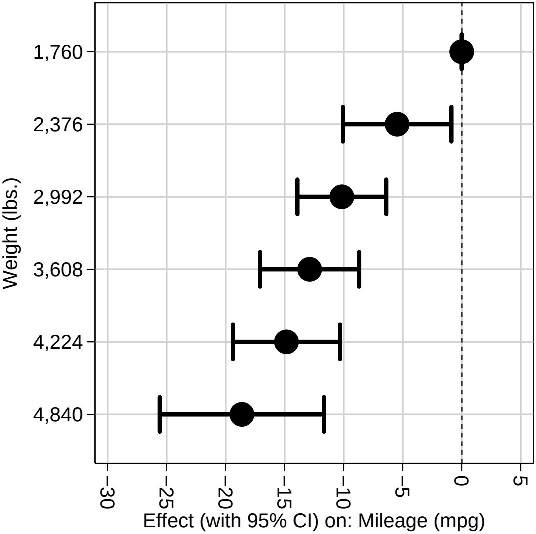
Effects of `weight` on `mpg` in the `auto` data

- The *Y*–axis displays the 6 reference levels of `weight`.
- The *X*–axis displays the effects of each weight on mileage, compared to the base weight of 1,760 pounds.
- *So*, once again, the reference splines model a continuous factor, just as dummy variables model discrete factors.



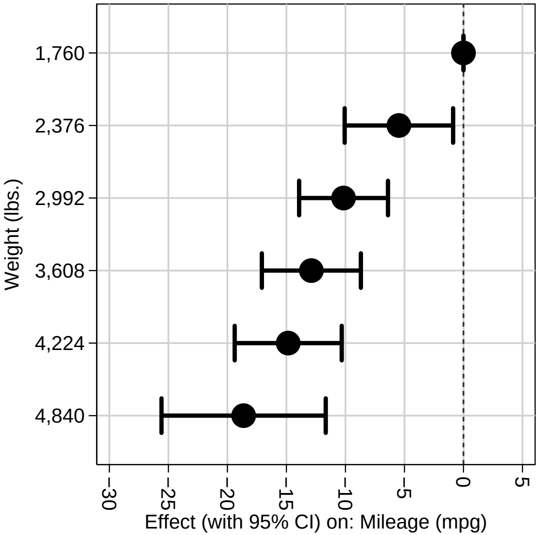
Effects of `weight` on `mpg` in the `auto` data

- The *Y*–axis displays the 6 reference levels of `weight`.
- The *X*–axis displays the effects of each weight on mileage, compared to the base weight of 1,760 pounds.
- *So*, once again, the reference splines model a continuous factor, just as dummy variables model discrete factors.



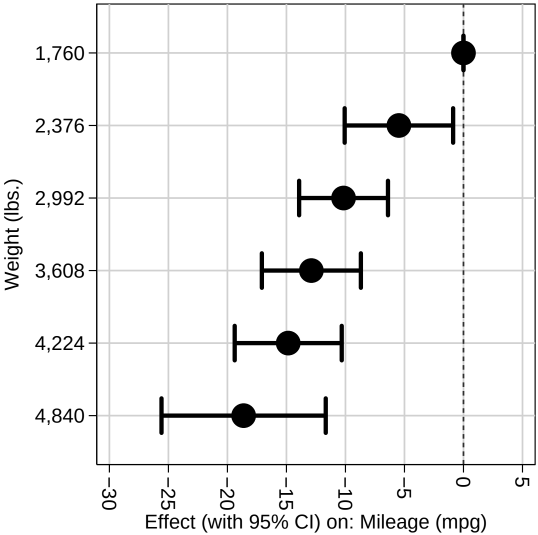
Effects of `weight` on `mpg` in the `auto` data

- The *Y*–axis displays the 6 reference levels of `weight`.
- The *X*–axis displays the effects of each weight on mileage, compared to the base weight of 1,760 pounds.
- *So*, once again, the reference splines model a continuous factor, just as dummy variables model discrete factors.



Effects of `weight` on `mpg` in the `auto` data

- The *Y*–axis displays the 6 reference levels of `weight`.
- The *X*–axis displays the effects of each weight on mileage, compared to the base weight of 1,760 pounds.
- *So*, once again, the reference splines model a continuous factor, just as dummy variables model discrete factors.



Product bases for multi–factor models

- In models with multiple *discrete* factors, we model the conditional Y–variable means or effects of combinations of factor values by generating **product bases**.
- Dummy–variable bases (complete or incomplete) are generated for discrete factors by `xi:` in Stata Version 10, and (in virtual form) by factor *varlist*s in Stata Versions 11 and 12.
- To combine discrete factors, we use the # operator in Stata 11 or 12 factor *varlist*s, or the $\star$ operator in `xi:` *varlist*s.
- Both of these work by generating product bases of binary dummy variables (possibly virtual), containing pairwise products of the dummy variables of the input bases.
- Similarly, given 2 or more reference–spline bases (complete or incomplete) for *continuous* factors, we can also generate product bases, with the same desirable features.
- The parameters corresponding to the product variables can be conditional means for factor–value combinations, subset–specific factor effects, or even "factor interaction effects".

Product bases for multi–factor models

- In models with multiple *discrete* factors, we model the conditional *Y*–variable means or effects of combinations of factor values by generating **product bases**.
- Dummy–variable bases (complete or incomplete) are generated for discrete factors by `xi:` in Stata Version 10, and (in virtual form) by factor *varlist*s in Stata Versions 11 and 12.
- To combine discrete factors, we use the # operator in Stata 11 or 12 factor *varlist*s, or the $\star$ operator in `xi:` *varlist*s.
- Both of these work by generating product bases of binary dummy variables (possibly virtual), containing pairwise products of the dummy variables of the input bases.
- Similarly, given 2 or more reference–spline bases (complete or incomplete) for *continuous* factors, we can also generate product bases, with the same desirable features.
- The parameters corresponding to the product variables can be conditional means for factor–value combinations, subset–specific factor effects, or even "factor interaction effects".

Product bases for multi–factor models

- In models with multiple *discrete* factors, we model the conditional Y–variable means or effects of combinations of factor values by generating **product bases**.
- Dummy–variable bases (complete or incomplete) are generated for discrete factors by `xi:` in Stata Version 10, and (in virtual form) by factor *varlist*s in Stata Versions 11 and 12.
- To combine discrete factors, we use the # operator in Stata 11 or 12 factor *varlist*s, or the $\star$ operator in `xi:` *varlist*s.
- Both of these work by generating product bases of binary dummy variables (possibly virtual), containing pairwise products of the dummy variables of the input bases.
- Similarly, given 2 or more reference–spline bases (complete or incomplete) for *continuous* factors, we can also generate product bases, with the same desirable features.
- The parameters corresponding to the product variables can be conditional means for factor–value combinations, subset–specific factor effects, or even "factor interaction effects".

Product bases for multi–factor models

- In models with multiple *discrete* factors, we model the conditional Y–variable means or effects of combinations of factor values by generating **product bases**.
- Dummy–variable bases (complete or incomplete) are generated for discrete factors by `xi:` in Stata Version 10, and (in virtual form) by factor *varlist*s in Stata Versions 11 and 12.
- To combine discrete factors, we use the # operator in Stata 11 or 12 factor *varlist*s, or the $\star$ operator in `xi:` *varlist*s.
- Both of these work by generating product bases of binary dummy variables (possibly virtual), containing pairwise products of the dummy variables of the input bases.
- Similarly, given 2 or more reference–spline bases (complete or incomplete) for *continuous* factors, we can also generate product bases, with the same desirable features.
- The parameters corresponding to the product variables can be conditional means for factor–value combinations, subset–specific factor effects, or even "factor interaction effects".

Product bases for multi–factor models

- In models with multiple *discrete* factors, we model the conditional Y–variable means or effects of combinations of factor values by generating **product bases**.
- Dummy–variable bases (complete or incomplete) are generated for discrete factors by `xi:` in Stata Version 10, and (in virtual form) by factor *varlist*s in Stata Versions 11 and 12.
- To combine discrete factors, we use the `#` operator in Stata 11 or 12 factor *varlist*s, or the $\star$ operator in `xi:` *varlist*s.
- Both of these work by generating product bases of binary dummy variables (possibly virtual), containing pairwise products of the dummy variables of the input bases.
- Similarly, given 2 or more reference–spline bases (complete or incomplete) for *continuous* factors, we can also generate product bases, with the same desirable features.
- The parameters corresponding to the product variables can be conditional means for factor–value combinations, subset–specific factor effects, or even "factor interaction effects".

Product bases for multi–factor models

- In models with multiple *discrete* factors, we model the conditional Y–variable means or effects of combinations of factor values by generating **product bases**.
- Dummy–variable bases (complete or incomplete) are generated for discrete factors by `xi:` in Stata Version 10, and (in virtual form) by factor *varlist*s in Stata Versions 11 and 12.
- To combine discrete factors, we use the `#` operator in Stata 11 or 12 factor *varlist*s, or the $\star$ operator in `xi:` *varlist*s.
- Both of these work by generating product bases of binary dummy variables (possibly virtual), containing pairwise products of the dummy variables of the input bases.
- Similarly, given 2 or more reference–spline bases (complete or incomplete) for *continuous* factors, we can also generate product bases, with the same desirable features.
- The parameters corresponding to the product variables can be conditional means for factor–value combinations, subset–specific factor effects, or even "factor interaction effects".

Product bases for multi–factor models

- In models with multiple *discrete* factors, we model the conditional *Y*–variable means or effects of combinations of factor values by generating **product bases**.
- Dummy–variable bases (complete or incomplete) are generated for discrete factors by `xi:` in Stata Version 10, and (in virtual form) by factor *varlist*s in Stata Versions 11 and 12.
- To combine discrete factors, we use the `#` operator in Stata 11 or 12 factor *varlist*s, or the `*` operator in `xi:` *varlist*s.
- Both of these work by generating product bases of binary dummy variables (possibly virtual), containing pairwise products of the dummy variables of the input bases.
- Similarly, given 2 or more reference–spline bases (complete or incomplete) for *continuous* factors, we can also generate product bases, with the same desirable features.
- The parameters corresponding to the product variables can be conditional means for factor–value combinations, subset–specific factor effects, or even "factor interaction effects".

flexcurv's team–mates: prodvars and fvprevar

- The prodvars package, downloadable from SSC, inputs 2 lists of input variables and generates a list of output variables, containing all pairwise products of the input variables.
- These output variables are named (and labelled) using rules specified by the user.
- The fvprevar package, downloadable from SSC, is a modified version of fvrevar.
- fvprevar inputs a Stata Version 11 or 12 factor *varlist*, and expands it, creating permanent output dummy variables instead of temporary output dummy variables.
- These permanent output dummy variables can be input to prodvars, together with reference splines, allowing the user to combine discrete and continuous factors at will.

flexcurv's team–mates: prodvars and fvprevar

- The prodvars package, downloadable from SSC, inputs 2 lists of input variables and generates a list of output variables, containing all pairwise products of the input variables.
- These output variables are named (and labelled) using rules specified by the user.
- The fvprevar package, downloadable from SSC, is a modified version of fvrevar.
- fvprevar inputs a Stata Version 11 or 12 factor *varlist*, and expands it, creating permanent output dummy variables instead of temporary output dummy variables.
- These permanent output dummy variables can be input to prodvars, together with reference splines, allowing the user to combine discrete and continuous factors at will.

`flexcurv`'s team–mates: `prodvars` and `fvprevar`

- The `prodvars` package, downloadable from SSC, inputs 2 lists of input variables and generates a list of output variables, containing all pairwise products of the input variables.
- These output variables are named (and labelled) using rules specified by the user.
- The `fvprevar` package, downloadable from SSC, is a modified version of `fvrevar`.
- `fvprevar` inputs a Stata Version 11 or 12 factor *varlist*, and expands it, creating permanent output dummy variables instead of temporary output dummy variables.
- These permanent output dummy variables can be input to `prodvars`, together with reference splines, allowing the user to combine discrete and continuous factors at will.

`flexcurv`'s team–mates: `prodvars` and `fvprevar`

- The `prodvars` package, downloadable from SSC, inputs 2 lists of input variables and generates a list of output variables, containing all pairwise products of the input variables.
- These output variables are named (and labelled) using rules specified by the user.
- The `fvprevar` package, downloadable from SSC, is a modified version of `fvrevar`.
- `fvprevar` inputs a Stata Version 11 or 12 factor *varlist*, and expands it, creating permanent output dummy variables instead of temporary output dummy variables.
- These permanent output dummy variables can be input to `prodvars`, together with reference splines, allowing the user to combine discrete and continuous factors at will.

`flexcurv`'s team–mates: `prodvars` and `fvprevar`

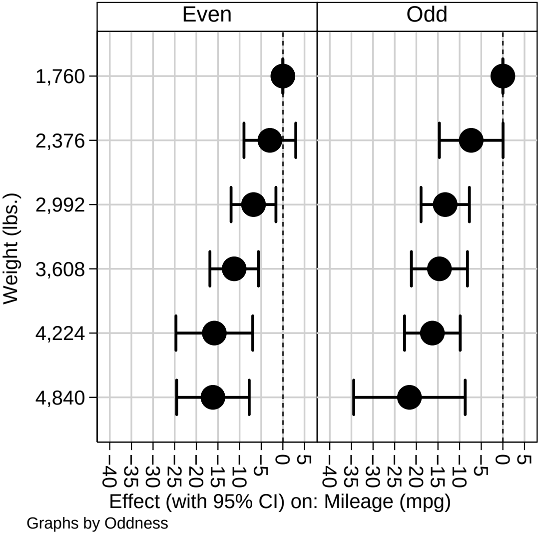
- The `prodvars` package, downloadable from SSC, inputs 2 lists of input variables and generates a list of output variables, containing all pairwise products of the input variables.
- These output variables are named (and labelled) using rules specified by the user.
- The `fvprevar` package, downloadable from SSC, is a modified version of `fvrevar`.
- `fvprevar` inputs a Stata Version 11 or 12 factor *varlist*, and expands it, creating permanent output dummy variables instead of temporary output dummy variables.
- These permanent output dummy variables can be input to `prodvars`, together with reference splines, allowing the user to combine discrete and continuous factors at will.

`flexcurv`'s team–mates: `prodvars` and `fvprevar`

- The `prodvars` package, downloadable from SSC, inputs 2 lists of input variables and generates a list of output variables, containing all pairwise products of the input variables.
- These output variables are named (and labelled) using rules specified by the user.
- The `fvprevar` package, downloadable from SSC, is a modified version of `fvrevar`.
- `fvprevar` inputs a Stata Version 11 or 12 factor *varlist*, and expands it, creating permanent output dummy variables instead of temporary output dummy variables.
- These permanent output dummy variables can be input to `prodvars`, together with reference splines, allowing the user to combine discrete and continuous factors at will.

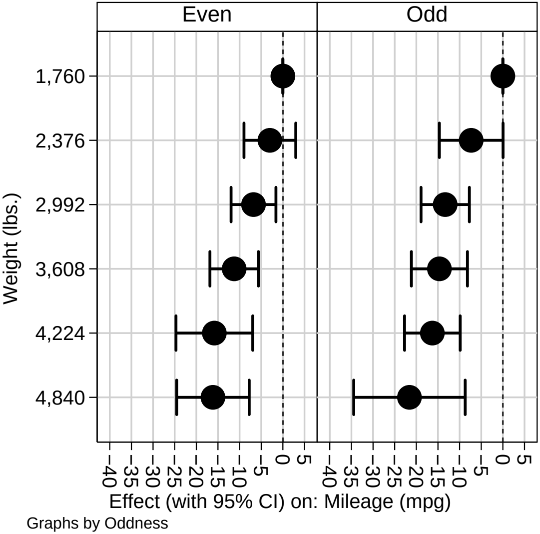
Example: Effects of `weight` on `mpg` in odd– and even–numbered models

- These confidence intervals are from an equal–variance regression model, combining weight as a continuous factor with oddness as a discrete factor.
- They were produced using `flexcurv`, `fvprevar` and `prodvars`.
- (We also used the SSC packages `parmest`, `descsave`, `fvregen`, `factext` and `eclplot` to make the plot.)



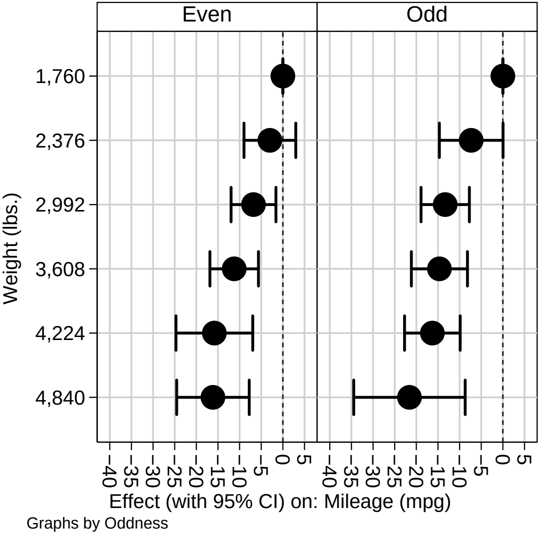
Example: Effects of `weight` on `mpg` in odd– and even–numbered models

- These confidence intervals are from an equal–variance regression model, combining weight as a continuous factor with oddness as a discrete factor.
- They were produced using `flexcurv`, `fvprevar` and `prodvars`.
- (We also used the SSC packages `parmest`, `descsave`, `fvregen`, `factext` and `eclplot` to make the plot.)



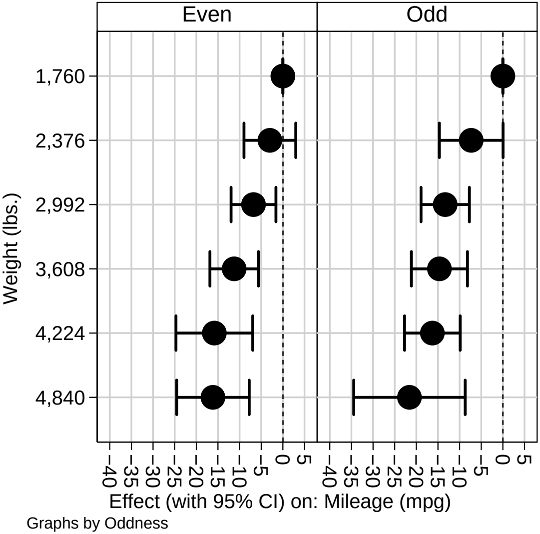
Example: Effects of `weight` on `mpg` in odd– and even–numbered models

- These confidence intervals are from an equal–variance regression model, combining weight as a continuous factor with oddness as a discrete factor.
- They were produced using `flexcurv`, `fvprevar` and `prodvars`.
- (We also used the SSC packages `parmest`, `descsave`, `fvregen`, `factext` and `eclplot` to make the plot.)



Example: Effects of `weight` on `mpg` in odd– and even–numbered models

- These confidence intervals are from an equal–variance regression model, combining weight as a continuous factor with oddness as a discrete factor.
- They were produced using `flexcurv`, `fvprevar` and `prodvars`.
- (We also used the SSC packages `parmest`, `descsave`, `fvregen`, `factext` and `eclplot` to make the plot.)



References

[1] Kandel, A. and S. C. Lee. 1979. *Fuzzy Switching and Automata: Theory and Applications*. London: Edward Arnold.

[2] Newson, R. 2000. sg151: *B*-splines and splines parameterized by their values at reference points on the *x*-axis. *Stata Technical Bulletin* **57**: 20–27.

[3] Newson, R. 2001. Splines with parameters that can be explained in words to non–mathematicians. Presented at the *7th UK Stata User Meeting*, 14 May, 2001. Downloadable from the conference website at *http://ideas.repec.org/s/boc/usug08.html*

[4] Schoenberg I. J. (ed.). 1969. *Approximations with Special Emphasis on Spline Functions*. New York: Academic Press.

This presentation, and the do–files producing the examples, can be downloaded from the conference website at *http://ideas.repec.org/s/boc/usug11.html*

The packages used in this presentation can be downloaded from SSC, using the `ssc` command.