StartR Splines and Smoothers

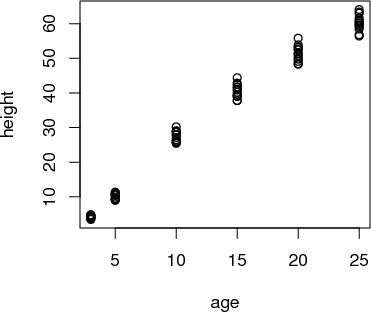
Mathematical models attempt to capture patterns in the real world. This is useful because the models can be more easily studied and manipulated than the world itself. One of the most important uses of functions is to reproduce or capture or model the patterns that appear in data.

Sometimes, the choice of a particular form of function — exponential or power-law, say — is motivated by an understanding of the processes involved in the pattern that the function is being used to model. But other times, all that’s called for is a function that follows the data and that has other desirable properties, for example is smooth or increases steadily.

“Smoothers” and “splines” are two kinds of general-purpose functions that can capture patterns in data, but for which there is no simple algebraic form. Creating such functions is remarkably easy, so long as you can free yourself from the idea that functions must always have formulas.

Smoothers and splines are defined not by algebraic forms and parameters, but by data and algorithms. To illustrate, consider some simple data. The data set Loblolly contains 84 measurements of the age and height of loblolly pines.

   pine = read.csv("Loblolly")  
   plot( height ~ age, data=pine, type="p")



Several three-year old pines of very similar height were measured and tracked over time: age five, age ten, and so on. The trees differ from one another, but they are all pretty similar and show a simple pattern: linear growth at first which seems to low down over time.

It might be interesting to speculate about what sort of algebraic function the loblolly pines growth follows, but any such function is just a model. For many purposes, measuring how the growth rate changes as the trees age, all that’s needed is a smooth function that looks like the data. Let’s consider two:

* A “cubic spline”, which follows the groups of data points and curves smoothly and gracefully.

   f1 = spliner(height ~ age, data=pine)

* A “linear interpolant”, which connects the groups of data points with straight lines.

   f2 = connector(height ~ age, data=pine)

The definitions of these functions may seem strange at first — they are entirely defined by the data: no parameters! Nonetheless, they are genuine functions and can be worked with like other functions. For example, you can put in an input and get an output:

   f1(age=8)

  [1] 20.68193

   f2(age=8)

  [1] 20.54729

You can graph them:

   plot(pine$age,pine$height)

plotFun(f1(age) ~ age, agelim=range(0,30))   
   plotFun(f2(age) ~ age, add=TRUE, col="red")  
   points(height ~ age, data=pine)

**Bonus**: Can you make a single graph with both the splines and the data points?

You can even “solve” them, for instance finding the age at which the height will be 35 feet:

   findZeros(f1(age) - 35 ~ age, xlim=range(0,30))

  [1] 12.69051

   findZeros(f2(age) - 35 ~ age, xlim=range(0,30))

  [1] 12.88436

In all respects, these are perfectly ordinary functions. All respects but one: there is no simple formula for them. You’ll notice this if you ever try to look at the computer-language definition of the functions:

   f2

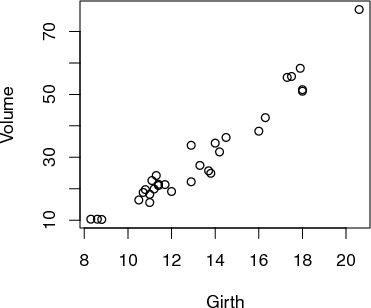
  function (age)  
  {  
      x = get(fnames[2])  
      if (connect)  
          SF(x)   
      else SF(x, deriv = deriv)  
  }  
  <environment: 0x10359d780>

There’s almost nothing here to tell you what the function is. The definition refers to the data itself which has been stored in an “environment.” These are computer-age functions, not functions from the age of algebra.

As you can see, the spline and linear connector functions are quite similar, except for the range of inputs outside of the range of the data. Within the range of the data, however, both types of functions go exactly through the center of each age-group.

Splines and connectors are not always what you will want, especially when the data are not divided into discrete groups, as with the loblolly pine data. For instance, the trees data set is measurements of the volume, girth, and height of black cherry trees. The trees were felled for their wood, and the interest in making the measurements was to help estimate how much usable volume of wood can be gotten from a tree, based on the girth (that is, circumference) and height. This would be useful, for instance, in estimating how much money a tree is worth. However, unlike the loblolly pine data, the black cherry data does not involve trees falling nicely into defined groups.

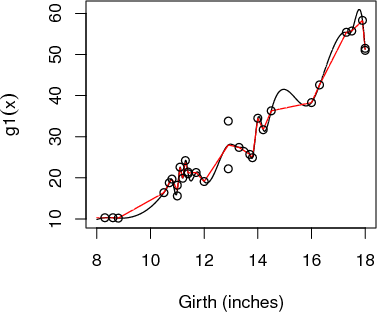
   cherry = read.csv("trees")  
   plot(Volume ~ Girth, data=cherry)



It’s easy enough to make a spline or a linear connector:

   g1 = spliner(Volume ~ Girth, data=cherry)  
   g2 = connector(Volume ~ Girth, data=cherry)   
   plotFun(g1(x) ~ x, xlim=range(8,18), xlab="Girth (inches)")   
   plotFun(g2(x) ~ x, add=TRUE, col="red")  
   points(Volume ~ Girth, data=cherry) # This does not seem to work, so again

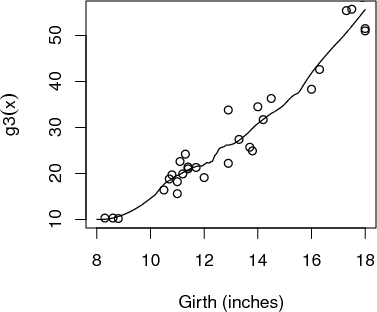
# Bonus if you get the following picture



The two functions both follow the data ... but a bit too faithfully! Each of the functions insists on going through every data point. (The one exception is the two points with girth of 13 inches. There’s no function that can go through both of the points with girth 13, so the functions split the difference and go through the average of the two points.)

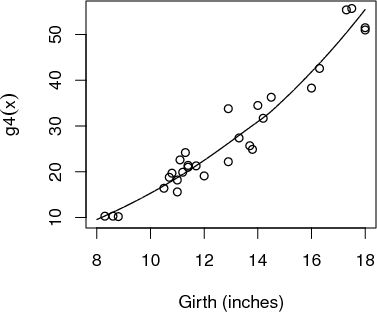
The up-and-down wiggling is of the functions is hard to believe. For such situations, where you have reason to believe that a smooth function is more appropriate than one with lots of ups-and-downs, a different type of function is appropriate: a smoother.

   g3 = smoother(Volume ~ Girth, data=cherry,span=0.5)   
   plotFun(g3(x) ~ x, xlim=range(8,18), xlab="Girth (inches)")   
   points(Volume~Girth, data=cherry)



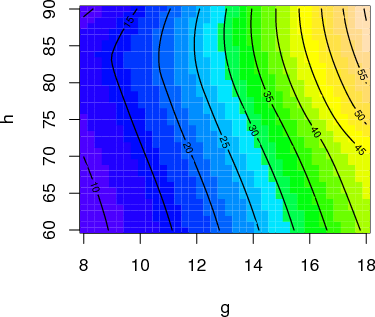
Smoothers are well named: they construct a smooth function that goes close to the data. You have some control over how smooth the function should be. The parameter span governs this:

   g4 = smoother(Volume ~ Girth, data=cherry,span=1.0)   
   plotFun(g4(x) ~ x, xlim=range(8,18), xlab="Girth (inches)")   
   points(Volume~Girth, data=cherry)



Of course, often you will want to capture relationships where there is more than one variable as the input. Smoothers do this very nicely; just specify which variables are to be the inputs.

   g5 = smoother(Volume ~ Girth+Height, data=cherry,span=1.0)   
   plotFun(g5(g,h) ~ g&h, glim=range(8,18), hlim=range(60,90))



When you make a smoother or a spline or a linear connector, remember these rules:

* You need a data frame that contains the data.
* You use the formula with the variable you want as the output of the function on the left side of the tilde, and the input variables on the right side.
* The function that is created will have input names that match the variables you specified as inputs. (For the present, only smoother will accept more than one input variable.)
* The smoothness of a smoother function can be set by the span argument. A span of 1.0 is typically pretty smooth. The default is 0.5.
* When creating a spline, you have the option of declaring monotonic=TRUE. This will arrange things to avoid extraneous bumps in data that shows a steady upward pattern or a steady downward pattern.

When you want to plot out a function, you need of course to choose a range for the input values. It’s often sensible to select a range that corresponds to the data on which the function is based. You can find this with the range( ) command, e.g.

   range(cherry$Height)

  [1] 63 87

**Exercises**

Exercise 1 The Orange data set

   orange = read.csv("Orange")  
   names(orange)

  [1] "Tree"          "age"           "circumference"

contains data on the circumference and age of trees.

(a)

Construct a smoother of circumference (in cm) as a function of age (in days), with span=1. What’s the value of the function for age 1000 days?

 127.41  128.32  130.28  132.08  133.12  138.20

(b)

Construct a smoother of circumference versus age with span=10. What’s the value for age 1000 days?

 127.41  128.32  130.28  132.08  133.12  138.20

(c)

How does the shape of the graph of the span=10 smoother differ from that of the span=1 smoother?

|  |  |
| --- | --- |
| A | The span=10 smoother is always above the span=1 smoother. |
| B | The span=10 smoother is always below the span=1 smoother. |
| C | The span=10 smoother is straighter than the span=1 smoother. |
|  |  |

Exercise 2 The cherry-tree data gives wood volume (in cubic feet) along with height (in feet) and girth (in inches — even though the name “Girth” is used, it’s really a diameter).

Construct a smoother of two variables for the cherry data with span=1. (Warning: Be careful to match the capitalization of the names).

What’s the value when height is 50 feet and “Girth” (that is, diameter) is 15 inches?

 24.4  25.5  26.6  27.7  28.8

Exericse 3 The data set BodyFat.csv contains several body measurements made on a group of 252 men. You’re going to look at theses variables:

* Weight — the total body weight in pounds.
* BodyFat — the percentage of total weight that is fat.
* Abdomen — the circumference at the waist in cm.

(a)

Construct a smoother of BodyFat as a function of Weight with span=1. What is the value of this function for Weight=150?

 13.46  14.36  16.34

What’s the general shape of the function?

|  |  |
| --- | --- |
| A | Sloping up and curved downwards |
| B | Sloping down and curved upwards |
| C | Straight line sloping downwards |
|  |  |

(b)

Construct a smoother of BodyFat as a function of Abdomen with span=1. What is the value of this function for Abdomen=100?

 23.34  23.85  24.43  25.23  26.74  27.25

What’s the general shape of the function?

|  |  |
| --- | --- |
| A | Sloping up and curved downwards |
| B | Sloping down and curved upwards |
| C | Straight line sloping downwards |
|  |  |

(c)

Construct a smoother of BodyFat as a function of both Abdomen and Weight with span=1. What is the value of this function for Abdomen=100 and Weight=150?

 23.34  23.85  24.43  25.23  26.74  27.25

Make a contour plot of your smoother function. Using the plot, answer this question: How does body fat percentage change when increasing weight from 140 to 240 pounds, but holding abdominal circumference constant at 90cm?

|  |  |
| --- | --- |
| A | Body fat percentage doesn’t change on weight. |
| B | Body fat percentage goes down with increasing weight. |
| C | Body fat percentage goes up with increasing weight. |
|  |  |