package: gsc 0.2.0

Generalised Shape Constraints (*Very* Rough Draft)

Charlotte Maia

December 2, 2010

This vignette introduces the gsc package, an R package for smoothing with generalised shape constraints. Here, a smooth function is regarded as a continuous function approximated by a series of points, and a shape constraint is regarded as a piece-wise constraint on the sign of a smooth function's zeroth, first, second or third derivative. One of the main goals of the package, is to generalise shape constraints, such that it's possible to specify an extensive variety of shapes. An initial (unconstrained) series is estimated using (locally-weighted) polynomials. Quadratic programming is used to transform the initial series, such that it satisfies the constraints.

Introduction

Smoothing with shape constraints, is the process of "smoothing", where there are constraints on the "shape" of a function. Here smoothing refers to estimating an unknown continuous function (from some data), by estimating a series of points, that approximate that function. Here shape refers to the sign of zeroth (positive/negative), first (increasing/decreasing), second (convex/concave) and third derivatives.

In principle, constrained smoothing here, is achieved in two steps, first an unconstrained smooth (via the ofp package), second a transformation. However, as the transformation can be applied to any regularly spaced series, in some cases we can transform a raw series (instead of a smooth series) or use a different smoother from the one used here.

A major goal of this package is to generalise such shape constraints. Hence, firstly we can apply multiple constraints (e.g. convex:increasing), and secondly, we can apply piece-wise constraints (i.e. specify one or more knots that partition a functions domain, and apply separate constraints to each segment of the domain).

Transformation is achieved using quadratic programming (via the quadprog package). The details of quadratic programming are hidden from the user, and shape constraints are specified in a reasonably intuitive form.

One problem with this approach is that transformation can result in smooth functions that aren't so smooth (i.e. have sharp angles in them). The author is considering adding smoothness (or roughness) constraints in the near future.

Generalised Shape Constraints

In the gsc package, constraints are represented by gsc objects. The constructor for a gsc object can take up to five arguments, first a numeric vector, giving the internal knots (i.e. x values that partition the domain), second a single (shape) string describing the signs of the zeroth derivative. The remaining arguments take the same form as the second argument, however apply to the first, second and third derivatives.

A shape string, can contain one character (a global constraint) or multiple characters (one character per segment). Characters can be a plus sign, for positive (strictly speaking, non-decreasing), a negative sign, for negative (strictly speaking, non-increasing), a zero, for constant, or a question mark for unconstrained.

For example, to create a gsc object with two knots (three segments), at ten and twenty, that's increasing (over all segments), convex over the first and third segments, and concave over the second segment, we can:

```
> #convex-concave-convex increasing
> g = gsc (c (10, 20), s1="+", s2="+-+")
```

Or a gsc object, with one knot (two segments), say at 1, constant over the first segment and unknown over the second segment.

```
> #constant-unknown
> g = gsc (1, s0="0?")
```

Or a gsc object with no knots (one global segment), that's positive and has a positive third derivative (effectively concave-convex).

```
> #(concave/convex) positive
> g = gsc (s0="+", s3="+")
```

Constrained Smoothing

This section describes a unified approach to constrained smoothing, that produces a smooth function. The next section describes the transformation itself, it deals with vectors more than functions, and can used with raw data, or with a separate smoother.

Given some x and y values:

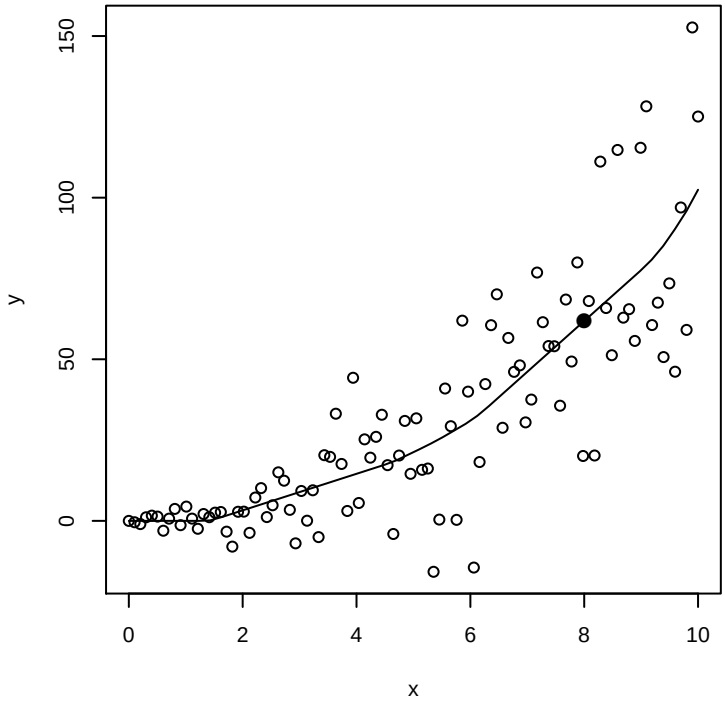
```
> n = 100
> x = seq (0, 10, length=n)
> y = x^2 + 3 * x * rnorm (n)
```

A constrained smooth (say convex:increasing:positive):

```
> g = gsc (s0="+", s1="+", s2="+")
> f = smoothc (g, x, y)
```

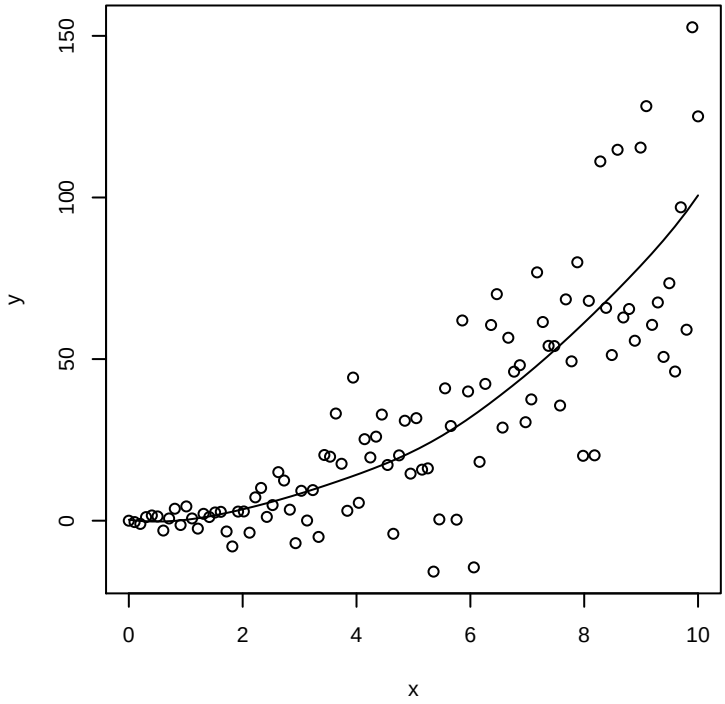
A plot (with a example point to demonstrate evaluation):

```
> plot (x, y)
> lines (f)
> points (8, f (8), pch=16, cex=1.5)
```



Another example, with re-smoothing:

```
> f = smoothc (g, x, y, smooth.out=TRUE)
> plot (x, y)
> lines (f)
```



Quadratic Programming Approach

Given an initial vector of y values, we can compute vector of transformed y values (say z values), that minimise the expression:

$$\sum_{\forall i}(z_i - y_i)^2$$

If there are no constraints, then the values of z should equal the values of y. Constraints in the form used in the earlier sections can be converted into a form suitable for quadratic programming, hence quadratic programming can be used for the transformation.

Given a gsc object, and a series of x values, and a series of y values (noting the y values can be almost any series, raw or smoothed), with can use the gsc fit method to compute transformed values of y.

```
> #note, not evaluated
> fit (g, x, y)
```