Smoothing Terms in GAM Models

John Maindonald

August 29, 2010

Contents

1 Generalized Additive Models (GAMs)

1.1 Introduction

In the account that will be given here, Generalized Additive Models (GAMs) extend linear and generalized linear models to include smooth functions of explanatory variables with the smoothness determined by either

a parameter that directly controls the smoothness of the curve, or

estimated predictive accuracy.

In the present discussion, the chief attention will be on smoothing terms that are spline functions of a single explanatory variable. Such functions can themselves be constructed as linear combinations of spline basis terms.

The account will proceed as follows:

1. An account of regression splines, which work with cubic spline basis terms of chosen degree. For this, a linear combination of spline basis terms is chosen that gives a curve that best fits the data.
2. The use of a basis that allows a high degree of flexibility in the chosen curve, but increasing the residual sum of squares by a roughness penalty that is some multiple λ of the integral of the squared first derivative.
   - Smoothing splines place a knot at each data point.
   - Penalized splines aim only to ensure that knots are well spread each data.
3. Use of generalized cross-validation (GCV) to determine the choice of λ.
4. The extension to generalized linear models (GLMs), in particular logistic regression models (for 0/1 data) and Poisson regression models (count data).

For an account of the detailed computations, see the document
`http://wwwmaths.anu.edu.au/%7Ejohnm/r-book/xtras/lm-compute.pdf`.
See Wood (2006) for a comprehensive account of GAM models as implemented in R's *mgcv* package.

1.2 Splines

A spline curve is a is piecewise polynomial curve, i.e., it joins two or more polynomial curves. The locations of the joins are known as "knots". In addition, there are boundary knots which can be located at or beyond the limits of the data. There are theoretical reasons for use of smoothly joining cubic splines. Smooth joining implies that the second derivatives agree at the knots where the curves join. Two types of splines are in common use:

Natural splines have zero second derivatives at the boundary knots. As a consequence, the curves extrapolate as straight lines.

B-splines are unconstrained at the boundary knots,

Spline curves of any given degree can be formed as a linear combination of basis functions. The *splines* package has two functions that may be used to generate basis terms – `bs()` which generates B-spline basis terms, and `ns()` which generates natural spline basis terms. In either case there are many different choices of basis functions.

Natural splines will be the major focus of attention. Let $g(x)$ be an arbitrary function that is formed from k cubic curves that join smoothly, with zero second derivatives at the boundary knots. Then there exists a basis $\phi_1(x), \phi_2(x), \ldots, \phi_k(x)$, such that:

$$g(x) = b_0 + b_1\phi_1(x) + b_2\phi_2(x) + \ldots + b_k\phi_k(x)$$

The basis terms span a vector space that has, after allowing one degree of freedom for the constant term, k degrees of freedom. If, instead, $\phi_1(x), \phi_2(x), \ldots, \phi_k(x)$ is a B-spline basis, the basis spans a vector space that, after allowing for the constant term, has $k+2$ degrees of freedom.

The following uses the abilities of the *splines* package, with data from the data frame `fruitohms` in the *DAAG* package. First `ohms` is plotted against `juice`. The function `ns()` (*splines* package) is then used to set up the basis functions for a natural spline with 3 degrees of freedom (`ns(juice, 3)`) and fit the curve. Figure 1 shows the plot:

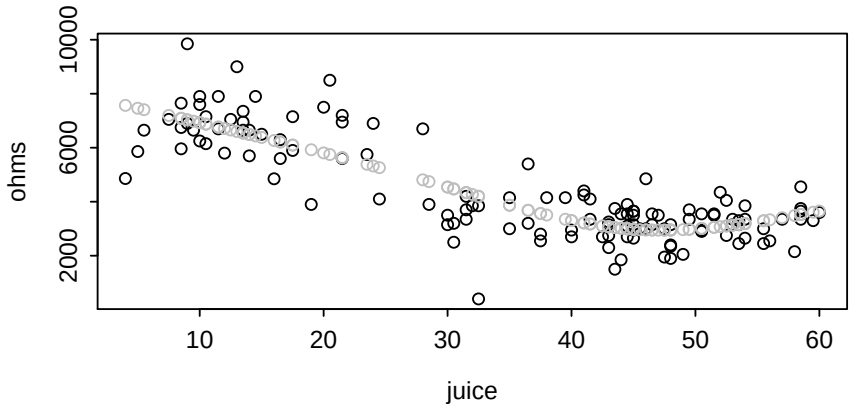

Figure 1: Smoooth curve, with pointwise 95% CI limits, fitted to the plot of resistance against apparent juice content. A three degree of freedom natural spline basis was used.

```
> library(DAAG)
> plot(ohms ~ juice, data=fruitohms)
> library(splines)
> fitohms <- fitted(lm(ohms ~ ns(juice, df=3), data=fruitohms))
> points(fitohms ~ juice, data=fruitohms, col="gray")
```

The parameter `df` (degrees of freedom) controls the smoothness of the curve. A large `df` allows a very flexible curve, e.g., a curve that can have multiple local maxima and minima. Clearly. the choice of a 3 degree of freedom curve, rather than 2 or 4, was arbitrary. The later discussion of Generalized Additive (GAM) models in Subsection 1.4 will show how this arbitrariness in choice of smoothing parameter can be avoided, providing data values can be assumed independent.

The advantage of regression splines is that they stay within the linear model (`lm()`) framework, with the same linear model theory and computational methods as any other linear model.

The `termplot()` function can be used to assess the result of a regression spline fit, just as for any other linear model fit. There is an option that allows, also, one standard error limits about the curve:

```
> ohms.lm <- lm(ohms ~ ns(juice, df=3), data=fruitohms)
> termplot(ohms.lm, partial=TRUE, se=TRUE)
```

The labeling on the vertical axis shows differences from the overall mean of `ohms`. In this example the *partial* is just the difference from the overall mean.

Natural spline basis functions, and their contributions to the fit

Figure 2A shows basis functions, both for natural splines of degree 3 (dashed curves) and of degree 4 (solid curves). Here, knots are placed at points that are equally spaced through the data. Notice that, in moving from a degree 3 natural spline curve to a degree 4 natural spline curve, the basis functions all change.

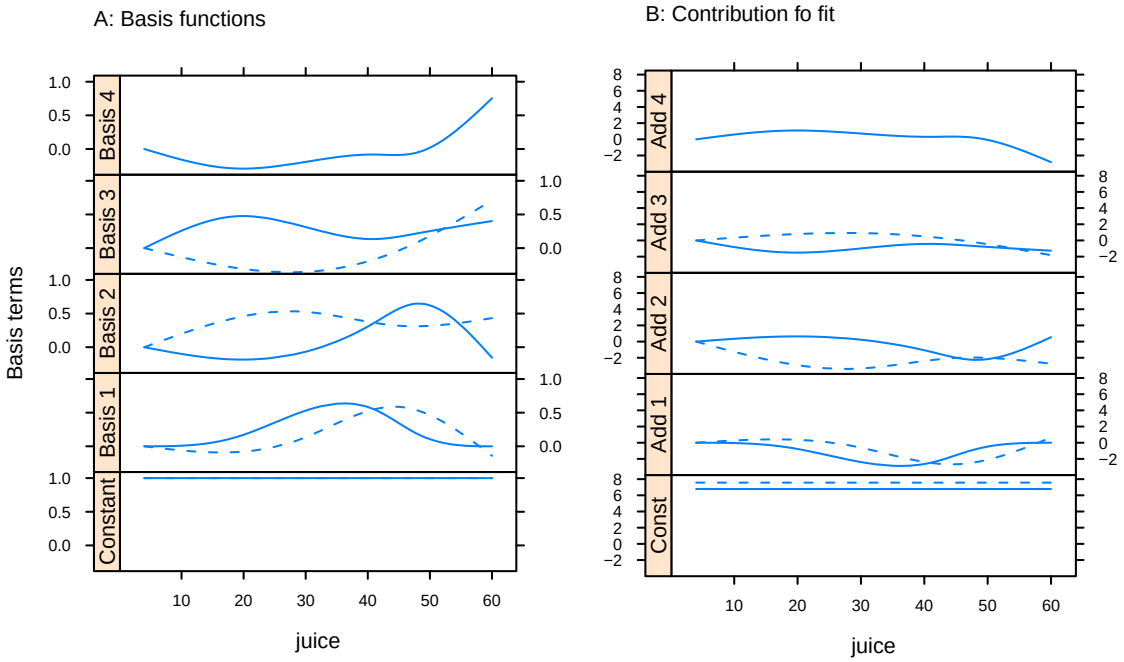

Figure 2: Panel A shows natural spline basis functions: (i) for a natural cubic spline of degree 3, with knots at the 33.3% and 66.7% quantile of the data (dashed curves); and (ii) for a natural cubic spline of degree 4, with knots at the 25%, 50% and 75% quantile of the data. Panel B shows the contributions of the basis functions to the fitted natural spline curve, in the regression of `kiloohms` on `juice`.

The two sets of basis functions can be extracted thus:

```
> matohms3 <- model.matrix(with(fruitohms, ~ ns(juice, 3)))
> matohms4 <- model.matrix(with(fruitohms, ~ ns(juice, 4)))
```

Spline basis elements

It is insightful to extract and plot the elements of the B-spline basis. Suitable code is:

```
> par(mfrow=c(2,2))
> basismat <- model.matrix(ohms.lm)
> for (j in 2:4) plot(fruitohms$juice, basismat[,j])
```

The first column of the model matrix is the constant term in the model. Remaining columns are the spline basis terms. The fitted values are determined by adding a linear combination of these four curves to the constant term.

Splines in models with multiple terms

For present purposes, it will be enough to note that this is possible. Consider for example

```
> loghills2k <- log(hills2000[, ])
> names(loghills2k) <- c("ldist", "lclimb", "ltime", "ltimef")
> loghill2k.lm <- lm(ltime ~ ns(ldist,2) + lclimb, data=loghills2k)
> par(mfrow=c(1,2))
> termplot(loghill2k.lm, col.term="gray", partial=TRUE,
           col.res="black", smooth=panel.smooth)
> par(mfrow=c(1,1))
```

1.3 Smooth functions of one explanatory variable

Smoothing spline smooths of a single variable place a knot at each data point. A penalty, some multiple λ of the integral of the squared second derivative of y with respect to x, is however added to the residual sum of squares, penalizing steep slopes. Consider a small interval δx over which the second derivative $f''(x)$ of the smoother $f(x)$ is approximately constant. The contribution of that interval to the penalty is then $\lambda f''(x)^2 \delta x$.

The total penalty is

$$\lambda \int f''(x)^2 dx$$

where the integral is over the range of x. The effect of the the penalty is to reduce the effective degrees of freedom. An adaptation of cross-validation – generalized cross-validation – is used to choose λ.

As noted above, the contributions of several variables can be added. There is then one λ_i, $i = 1, \ldots p$, for each of the p variables.

The placing of a knot at each data point has the result that the number of basis functions is one less than the number of data points. Even with just one explanatory variable, this can computationally expensive. A reasonable solution is to work, as in the penalized spline approach, with some smaller number of knots that are spread evenly through the data. Alternatively, use may be made of a low rank approximation to the space spanned by the complete set of basis terms.

A further challenge is to define and fit general smoothing functions of several explanatory variables. Thin plate splines are one approach. A set of basis functions emerges directly from the demand to minimize the residual sum of squares, plus a smoothness penalty that is a multiple λ of the multivariate integral over the space spanned by the explanatory variables of a suitable smoothness function.

These smoothers have as many parameters as there are unique predictor combinations, so that the computational cost is proportional to the cube of the number of of variables. Minimization over the full set of basis functions can be therefore computationally demanding, or may be intractable. Use of a low rank approximation to the space spanned by the thin plate spline basis may be essential. Wood (2006) calls the resulting basis a thin plate regression spline basis.

The approaches that are described generalize for use with the error terms that are available for GLM models, now working with a penalized likelihood. The function `gam()`, in the *mgcv* package, handles the fitting in a highly automatic manner.

1.4 GAM models with normal errors

Fitting a GAM model with a single smoothing term

In Figure 3, residuals were calculated from a linear regression of `log(Time)` on `log(Distance)`, with data from the `worldRecords` dataset in the *DAAG* package. Then the function `gam()` was used to pass a smooth curve through the residuals.

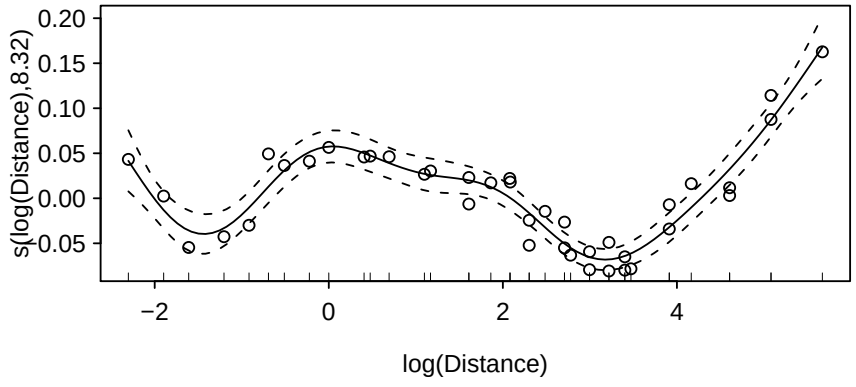

Figure 3: Residuals from the regression of `log(Time)` on `log(Distance)`, with smooth curve fitted. Data are from the `worldRecords` dataset (*DAAG* package).

Code that handles the calculations and plots the result is:

```
> library(mgcv)
> res <- resid(lm(log(Time) ~ log(Distance), data=worldRecords))
> wr.gam <- gam(res ~ s(log(Distance)), data=worldRecords)
> plot(wr.gam, residuals=TRUE, pch=1, las=1)
```

As `Time` is on a scale of natural logarithms, a residual of 0.1 corresponds to a deviation from the line of $\exp(0.1) - 1 \simeq 10.5\%$, i.e., just a little larger than 10%. A residual of 0.15 corresponds to a deviation from the line of ~16.2%. The magnitude of these deviations may seem surprising, given that the graph shows the points lying very close to the regression line of `log(Time)` on `log(Distance)`. The reason is that the times span a huge range, with the largest time more than 8800 × the smallest time. Set against a change by a factor of 8800, a change of 15% is very small.

If the preference is to fit a smooth curve to the initial data, the following code may be used:

```
> wrdata.gam <- gam(log(Time) ~ s(log(Distance)), data=worldRecords)
> plot(wrdata.gam, residuals=TRUE, pch=1)
```

On the graph that results, the 95% pointwise confidence bounds are hard to distinguish from the line.

With models such as these, it is good practice to check whether any consistent pattern appears with random data. As will be shown below, this can happen when the density of points varies along the x-axis, with the result that a single smoothing parameter is inappropriate.

Among the possibilities for creating "random" data are:

- Permute the residuals.
- Take a bootstrap sample of the residuals.
- Take random normal data from a population with mean 0 and standard deviation the same as that of the residuals.

Figure 4 shows the plots from 6 random permutations of the residuals.

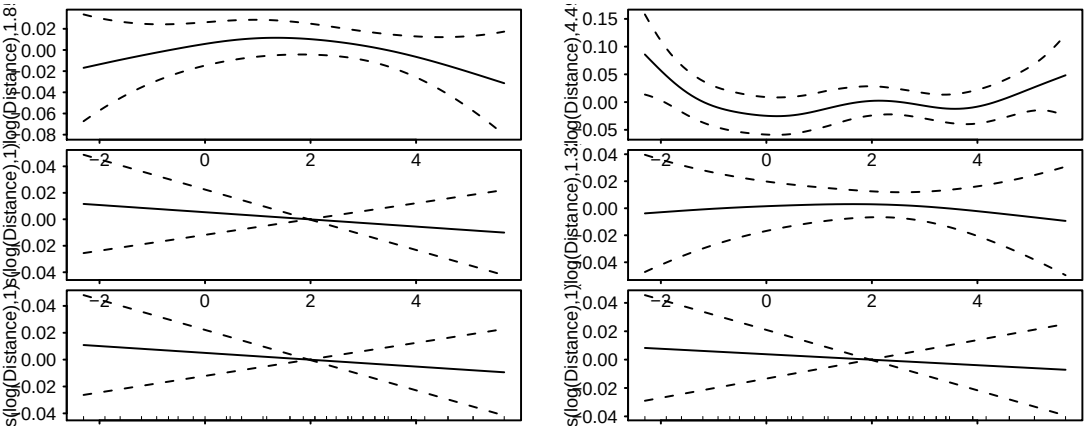

Figure 4: Plots from GAM models, fitted to successive random permutations of the residuals from the straight line fit of `log(Time)` to `log(Distance)`, for the `worldRecords` data. Note that none of thse plots shows more than a hint of a pattern.

Reassuringly, none of these show a pattern that stands out clearly, relative to the 95% pointwise error limits.The code used is:

```
> opar <- par(mfrow=c(3,2), mar=c(0.25, 4.1, 0.25, 1.1))
> res <- resid(lm(log(Time) ~ log(Distance), data=worldRecords))
```

```
> for(i in 1:6){
    permres <- sample(res)  # Random permutation
    # 0 for left-handers
    # 1 for right
    perm.gam <- gam(permres ~ s(log(Distance)), data=worldRecords)
    plot(perm.gam, las=1, rug=if(i<5)FALSE else TRUE)
  }
> par(opar)
```

Fitting a GAM model to climate data – two smooth terms

Time series data are likely to be correlated between years, creating potential issues for the use of smoothing methodology. In fortunate circumstances, any underlying trend will stand out above the error, but this should not be taken for granted. Simple forms of relationship are more plausible than complicated forms of relationship. The error term in regression relationships are used to explain synchrony between series is less likely to be less affected by autocorrelation than errors in the separate series. With these cautions, we proceeed to examination of a time series regression relationship.

Figure 5 fits annual rainfall, in the Murray-Darling basin of Australia, as a sum of smooth functions of `Year` and `SOI`. Figure 5 shows the estimated contributions of the two model terms.

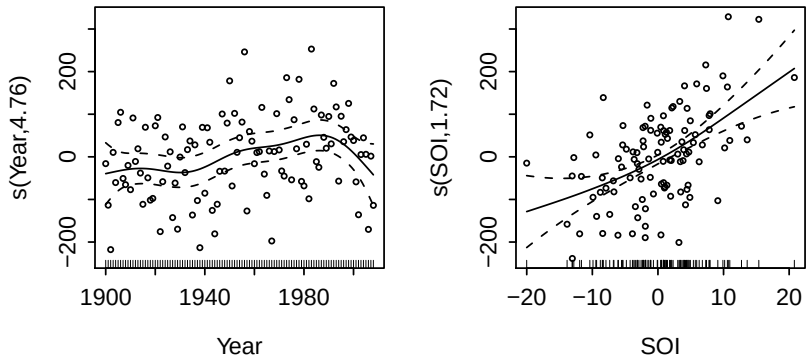

Figure 5: Contributions of the model terms to `mdbRain`, in a GAM model that is the sum of smooth terms in `Year` and `Rain`. The dashed curves show pointwise 2-SE limits, for the fitted curve. Note the downturn in the trend of `mdbRain` after about 1985.

Code is:

```
> par(mfrow=c(1,2))
> mdbRain.gam <- gam(mdbRain ~ s(Year) + s(SOI), data=bomregions)
> plot(mdbRain.gam, residuals=TRUE, se=2, pch=1, cex=0.5, select=1)
> plot(mdbRain.gam, residuals=TRUE, se=2, pch=1, cex=0.5, select=2)
> par(mfrow=c(1,1))
```

1.5 Smoothing terms with time series data – issues of interpretation

Now consider the data series `Erie`, giving levels of Lake Erie from 1918 to 2009 will be used for illustration.[1] They are available in versions ≥ 0.7-8 of the *DAAGxtras* package, as the column `Erie` in the multivariate time series object `greatLakes`. It is convenient to start by extracting the column that will be used:

```
> Erie <- greatLakes[,"Erie"]
```

An ideal would be to find a covariate or covariates than can largely explain the year to year changes. For the series that we now examine, no such explanation is available.

[1] Data are from `http://www.lre.usace.army.mil/greatlakes/hh/greatlakeswaterlevels/historicdata/greatlakeshydrographs/`

The unsmoothed data

Figure 6 shows a plot of the series:

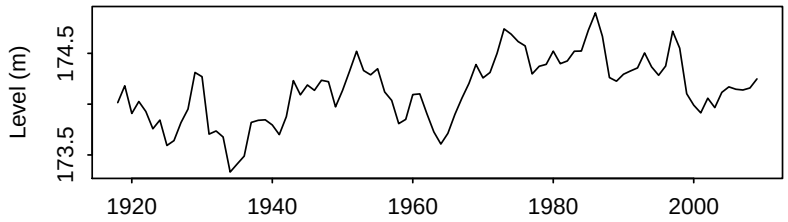

Figure 6: Level of Lake Erie, in meters.

```
> ## Code
> plot(Erie,
       xlab="",
       ylab="Level (m)")
```

In the absence of identfying a direct cause for the year to year changes, the best that can be done is to find a correlation structure that drives much of the year to year change. A re-run of the process (a new *realization*) will produce a different series, albeit one that shows the same general tendency to move up and down.

The plots that are show in Figure 7 are good starting points for investigation of the corrrelation structure. Panel A shows lag plots, up to a lag of 3. Panel B shows estimates of the successive correlations, in this context are called autocorrelations.

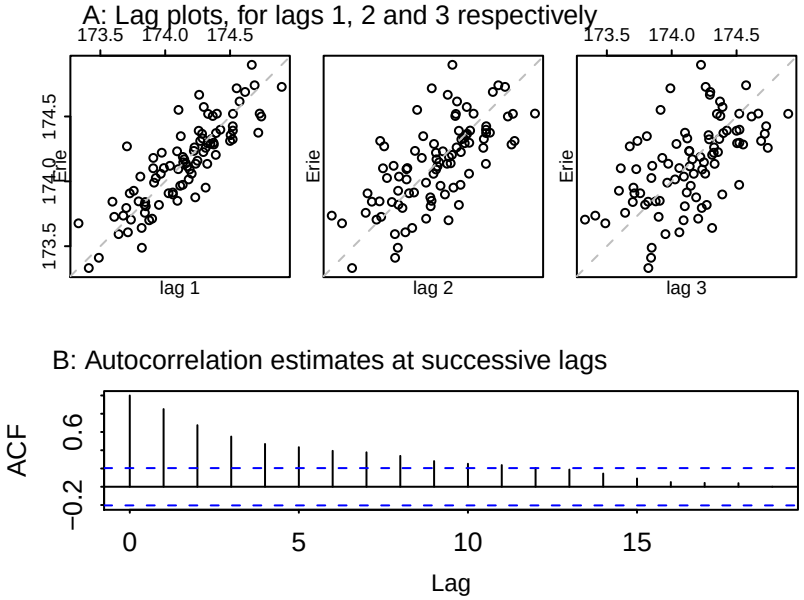

Figure 7: Panel A plots lake levels vs levels at lags 1, 2 and 3 respectively, for Lake Erie. Panel B shows the autocorrelations at lags 0 (= 1), 1 (1^{st} graph in panel A), 2 (2^{nd} graph), ... A large autocorrelation at lag 1 is followed by smaller autocorrelations at later lags.

```
> ## Panel A
> lag.plot(Erie, lags=3,
           do.lines=FALSE,
           layout=c(1,3))
> ## Panel B
> acf(Erie)
```

There is a strong correlation at lag 1, a strong but weaker correlation at lag 2, and a noticeable correlation at lag 3. Such a correlation pattern is typical of an autoregressive process where most of the sequential dependence can be explained as a flow-on effect from a dependence at lag 1.

In an autoregressive time series, an independent error component, or "innovation" is associated with each time point. For an order p autoregressive time series, the error for any time point is obtained by taking the innovation for that time point, and adding a linear combination of the innovations at the p previous time points. (For the present time series, initial indications are that $p = 1$ might capture most of the correlation structure.)

1.5.1 Smooth, with automatic choice of smoothing parameter

What do we see if we fit a GAM smoothing term to the `Erie` series? Recall that the smooth assumes independently and identically distributed data, and in particular that there is no serial correlation.

Figure 8 shows the result:

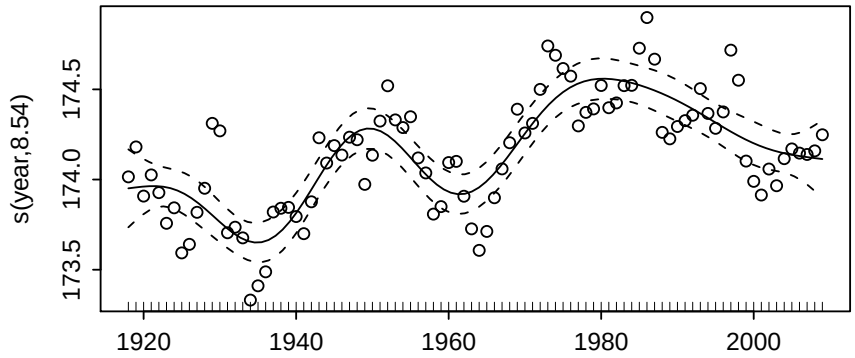

Figure 8: GAM smoothing term, fitted to the Lake Erie Data. The curve has removed the autocorrelation structure from the series, leaving residuals that are independent.

The code is:

```
> df <-  data.frame(height=as.vector(Erie), year=time(Erie))
> obj <- gam(height ~ s(year), data=df)
> plot(obj, shift=mean(df$height), residuals=T, pch=1, xlab="")
```

The curve may be useful as a broad summary of the pattern of change over time. The pointwise confidence limits would be meaningful if the processes that generated the sequential correlation structure could be identified and used to explain the curve. Without such an understanding, they are meaningless. All that is repeatable is the process that generated the curve, not the curve itself. Time series models, such as will now be considered, are designed to account for such processes.

Fitting and use of an autoregressive model

An autoregressive model gives insight that makes it possible to estimate the level of the lake a short time ahead, and to put realistic confidence bounds around those estimates. For the Lake Erie data, an autoregressive correlation structure doesw a good job of accounting for the pattern of change around a mean that stays constant.

Once an autoregressive model has been fitted, the function `forecast()` in the *forecast* package can be used to predict future levels, albeit with very wide confidence bounds.

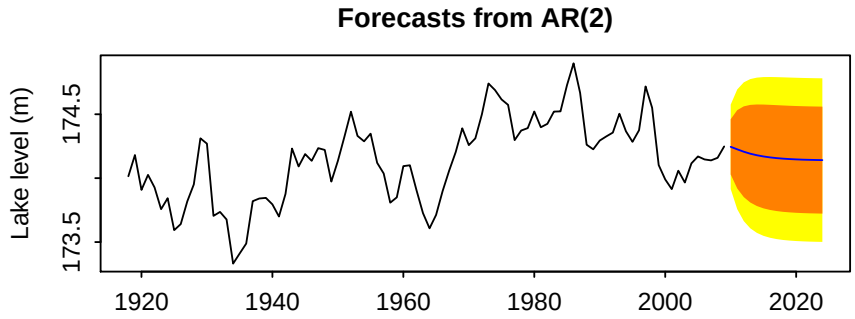

Figure 9: Predictions, 15 years into the future, of Lake Erie levels (m). The shaded areas give 80% and 95% confidence bounds.

The code is:

```
> erie.ar <- ar(Erie)
> library(forecast)
> plot(forecast(erie.ar, h=15), ylab="Lake level (m)")   # 15 time points ahead
```

To see the paramteters of the model that has been fitted, type:

```
> erie.ar
```

Fitting smooth curves to simulations of an autoregressive process

In order to reinforce the points just made, consider results from fitting smooth curves to repeated simulations of an autoregressive process, here with a lag 1 correlation of 0.7:

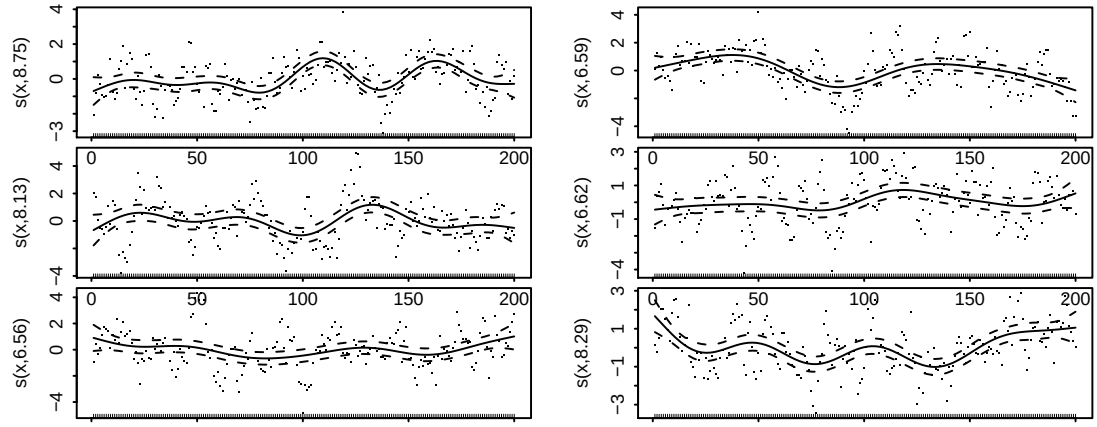

Figure 10:

Code for one of these plots is:

```
> df <- data.frame(x=1:200, y=arima.sim(list(ar=0.7), n=200))
> df.gam <- gam(y ~ s(x), data=df)
> plot(df.gam, residuals=TRUE)
```

The smooth curves, fitting assuming independent errors, are different on each occasion. They serve no useful purpose, if the aim is to generalize beyond the particular realization that generated them.

This brief excursion into a simple form of time series model is intended only to indicate the limitations of automatic smooths. Among several recent introductions to time series that include R code for the computations, note Hyndman et al. (2008), which is designed for use with the *forecasting* bundle of packages.

1.6 Logistic regression with GAM smoothing term

Several articles in medical journals have used data on first class cricketers in the UK to suggest that left-handers, i.e., cricketers who used their left hand for bowling, have a shorter life-span than right-handers. Those articles did not however account for changes over time in the proportions of left-handers. Similar studies have been done for basketballers, again ignoring systematic changes over time in the proportions of left-handers.

For cricketers born between 1840 and 1960, the total numbers are:

```
> ## The cricketer dataset is in the DAAG package
> table(cricketer$left)

right  left
 4859  1101
```

The proportion of left-handers is a little under 18.5%.

A GAM model, with binomial link, will show how the proportion may have changed. Here, the independence assumption is very plausible. There may be occasional father and son successions of left-handers, but these are likely to make only a very small contribution to the total data.

The following does the calculations

```
> library(mgcv)
> hand <- with(cricketer, as.vector(as.vector(unclass(left)-1)))
>   # 0 for left-handers
>   # 1 for right
> hand.gam <- gam(hand ~ s(year), data=cricketer, family=binomial)
```

Figure 11 plots the result:

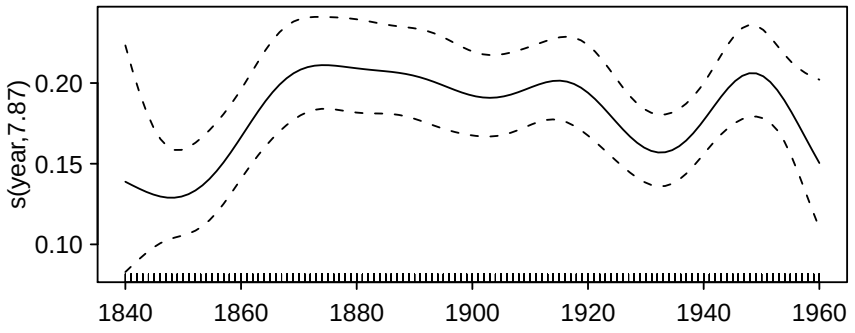

Figure 11: Plot from a GAM model in which the proportion of left-handed cricketers has been modeled as a smooth function of year of birth. The dashed lines are approximate two standard error limits.

The code that did the plotting is:

```
> plot(hand.gam, las=1, xlab="",
       trans=function(x)exp(x)/(1+exp(x)),
       shift=mean(predict(hand.gam)))
```

As a check, Figure 12 does several fits in which left-handers are generated by a random process, with a constant 18.5% proportion:

The code used is:

```
> opar <- par(mfrow=c(5,1), mar=c(0.25, 4.1, 0.25, 1.1))
> for(i in 1:5){
  hand <- sample(c(0,1), size=nrow(cricketer), replace=TRUE,
                 prob=c(0.185, 0.815))
    # 0 for left-handers
    # 1 for right
  hand.gam <- gam(hand ~ s(year),
                  data=cricketer)
  plot(hand.gam, las=1, xlab="",
       rug=if(i<5)FALSE else TRUE,
       trans=function(x)exp(x)/(1+exp(x)),
       shift=mean(predict(hand.gam)))
  }
> par(opar)
```

Occasionally, one or more of these plots will show an apparent pattern.

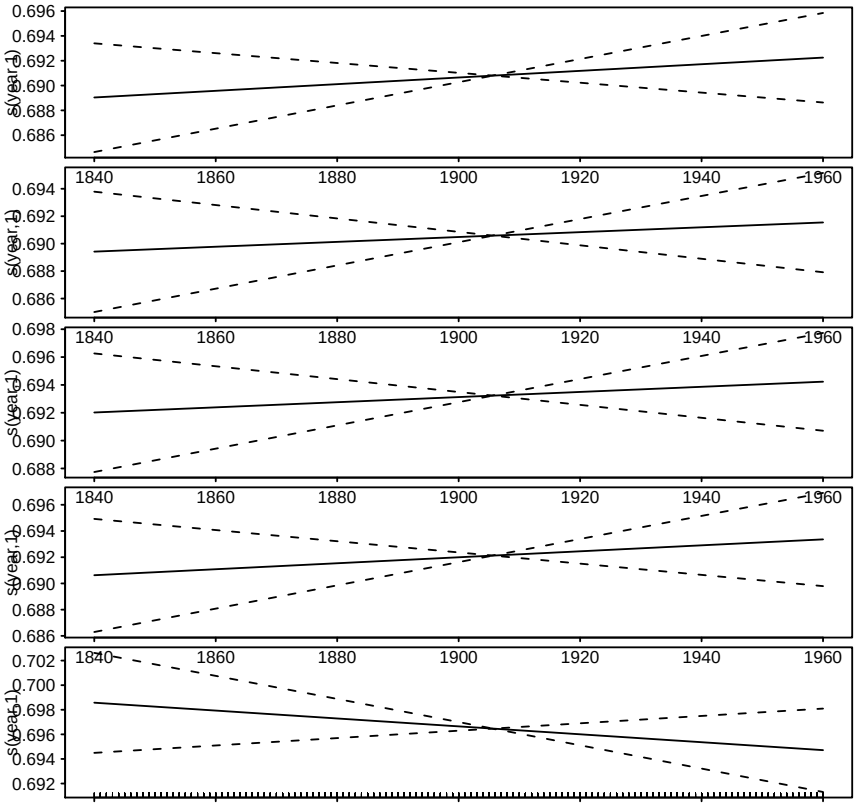

Figure 12: Plots from GAM models, fitted to successive random draws from a population in which the proportion of lefthanded cricketers is as constant 18.5%, irrespective of date of birth.

1.7 Poisson regression with GAM smoothing term

There may be some insight to be gained from changes in the numbers of left-handers and right-handers separately. For this, we assume that left-handers and right-handers are generated by separate Poisson processes, according to rates that vary smoothly over time. The numbers of left-handers and right-handers can be summed for each year, when these annual numbers also follow a Poisson process.

The following code fits the model:

```
> rtlef <- data.frame(with(cricketer, as(table(year, left),"matrix")))
> rtlef$year <- as.numeric(rownames(rtlef))
> denright <-  gam(right ~ s(year), data=rtlef, family=poisson)
> denleft <-  gam(left ~ s(year), data=rtlef, family=poisson)
> fitright <- predict(denright, type="response")
> fitleft <- predict(denleft, type="response")
```

Code that does the plotting, as in Figure 13 is:

```
> opar <- par(mar=c(2.1,4.6,3.1,0.6), mgp=c(2.65, .5,0))
> plot(fitright ~ year, col="blue", main="", type="n", xlab="",
       ylab="Number of cricketers\nborn in given year",
       ylim=c(0, max(rtlef$right)), data=rtlef)
> with(rtlef, lines(fitright ~ year, col="blue", lwd=2))
> with(rtlef, lines(fitleft ~ year, col="purple", lwd=2))
> with(rtlef, lines(I(4*fitleft) ~ year, col="purple", lty=2))
> legend("topleft", legend=expression("Right-handers", "Left-handers",
      "Left-handers "%*%" 4"),
         col=c("blue", "purple", "purple"), pch=c(1,1), lty=c(1,1,2),
         lwd=c(2,2, 1),bty="n", inset=0.01)
> par(opar)
```

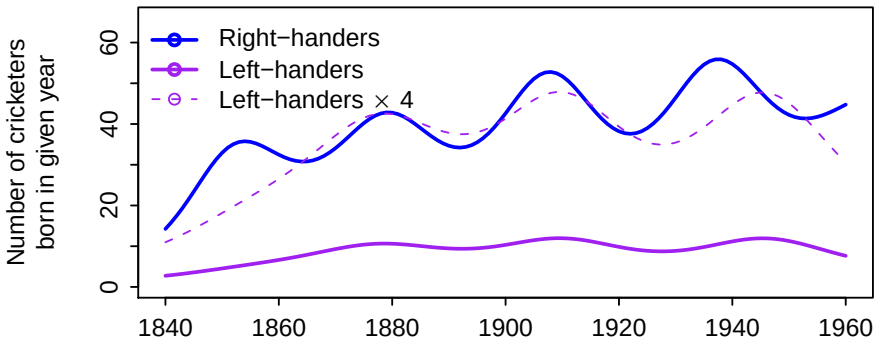

Figure 13: Numbers of left and right-handers have been separately modeled as smooth functions of year of birth. The dashed curve scales up the number of left-handers by a factor of 4, for easy comparison of the patterns of variation of the two curves.

Both curves can be fitted at once, thus:

```
> num.df <- with(cricketer, as.data.frame((table(year, left))))
> num.df$year <- as.numeric(as.character(num.df$year))
> num.gam <-  gam(Freq ~ left + s(year, by=left), data=num.df, family=poisson)
> par(mfrow=c(1,2))
> plot(num.gam)
> par(mfrow=c(1,1))
```

Observe that in `Freq ~ left + s(year, by=left)`, the factor `left` has appeared twice – giving separate offsets (constant terms) for the two curves, and separate patterns of variation about the respective offsets.

Counts that are quasipoisson

To do!

1.8 Exercises

1. Below, we start with an underlying curve to an underlying curve $mu = \frac{\sin x}{x}$ and add one or other of: (i) random normal noise, or (ii) autocorrelated random data. The underlying curve is:

   ```
   > x <- seq(from=-10, to =10, by=0.55)  ## NB: Best avoid x=0
   > mu <- sin(x)/x
   ```

   Consider the model simulations A and B

   ```
   > ## A: Add white noise (random normal noise) with mean 0, specify sd
   > makewhite <- function(mu, sd){
       mu + rnorm(n=length(mu), sd=sd)
     }
   > ## B: Add a series with mean 0, sd=0.2, autocorrelation 0.6
   > ## NB: sd is the standard deviation of the innovations.
   > makeAR1 <- function(mu, sd, ar=0.5){
       mu + arima.sim(list(ar=ar), n=length(mu), sd=sd)
     }
   ```

   The following plots the data for three realisations of each of series A and series B, then using the `gam()` function to fit a smooth curve and add the smooth to the plots:

   ```
   > opar <- par(mfrow=c(2,3))
   > for(i in 1:3){
   ```

```
    y <- makewhite(mu, sd=0.15)
    white.gam <- gam(y ~ s(x))
    plot(white.gam, resid=TRUE, shift=white.gam$coef[1],
         pch=1, se=FALSE, xlab="")
    if(i==1)mtext(side=3, line=0.4, "Add white noise")
  }
> for(i in 1:3){
    y1 <- makeAR1(mu, sd=0.15, ar=0.5)
    ar1.gam <- gam(y1 ~ s(x))
    plot(ar1.gam, resid=TRUE, shift=ar1.gam$coef[1],
         pch=1, se=FALSE, xlab="")
    if(i==1)mtext(side=3, line=0.4, "Add autoregressive 'noise'")
  }
> par(opar)
```

Repeat the plots with `sd=0.45` and with `sd=0.75`. Under what circumstances is the autocorrelated error least likely to distort the smooth? Under what circumstances is a spurious smooth likely?

2 References

References

Hyndman, R. J.; Koehler, A. B.; Ord, J. K.; and Snyder, R. D. 2008. *Forecasting with Exponential Smoothing: The State Space Approach*, 2^{nd} edn, Springer.

Wood, S. N. 2006. *Generalized Additive Models.* An Introduction with R. Chapman & Hall/CRC. [This has an elegant treatment of linear models and generalized linear models, as a lead-in to generalized additive models. It is an almost indispensible reference for use of the `mgcv` package for R. Refer to it for the theory of and practical use of regression splines, various types of penalized splines, thin plate splines, etc. A final chapter is devoted to Generalised Additive mixed models.]