MICHAEL CLARK
CENTER FOR SOCIAL RESEARCH
UNIVERSITY OF NOTRE DAME

GENERALIZED ADDITIVE MODELS

GETTING STARTED WITH ADDITIVE MODELS IN R

Contents

Preface

The following provides a brief introduction to generalized additive models and some thoughts on getting started within the R environment. It doesn't assume much more than a basic exposure to regression, and maybe a general idea of R though not necessarily any particular expertise. The presentation is of a very applied nature, and such that the topics build upon the familiar and generalize to the less so, with the hope that one can bring the concepts they are comfortable with to the new material. The audience in mind is a researcher with typical social science training, though is not specific to those disciplines alone.

To familiarize yourself with R, I have an introduction to R here and second part here, and they would provide all one would need for the following, though there is much to be gained from simple web browsing on R. And while it wasn't the intention starting out, this document could be seen as a vignette for the *mgcv* package.

With regard to the text, functions are in red text, packages in blue, and links are sort of orange. With regard to the graphs, they are meant to be zoomed in upon, and you can do so without loss of clarity. Having them smaller and in the margins allows the reader to focus on them as much or little as you like as one would like while having them right where they need to be in relation to the text (for the most part). If using Adobe reader, I usually have to turn off 'enhance thin lines' in preferences in order to see the graphs as they are originally constructed, but other readers should be fine.

This document was created entirely with Rstudio.

Last modified November 19, 2013.
Original draft August, 2012.

Introduction

Beyond the General Linear Model I

General Linear Model

LET US BEGIN by considering the standard ordinary least squares (OLS) linear regression model. We have some response/variable we wish to study, and believe it to be some function of other variables. In the margin to the right, y is the variable of interest, assumed to be normally distributed with mean μ and variance σ^2, and the Xs are the predictor variables/covariates in this scenario. The predictors are multiplied by the coefficients (beta) and summed, giving us the linear predictor, which in this case also directly provides us the estimated fitted values.

$$y \sim \mathcal{N}(\mu, \sigma^2)$$
$$\mu = b_0 + b_1X_1 + b_2X_2 ... + b_pX_p$$

And since we'll be using R and might as well get into that mindset, I'll give an example of how the R code might look like.

```
mymod = lm(y ~ x1 + x2, data = mydata)
```

One of the issues with this model is that, in its basic form it can be very limiting in its assumptions about the data generating process for the variable we wish to study. Also, while the above is one variant of the usual of the manner in which it is presented in introductory texts, it also very typically will not capture what is going on in a great many data situations.

Generalized Linear Model

In that light, we may consider the *generalized* linear model. Generalized linear models incorporate other types of distributions[1], and include a link function $g(.)$ relating the mean μ, or stated differently, the estimated fitted values $E(y)$, to the linear predictor $X\beta$, often denoted η. The general form is thus $g(\mu) = X\beta$.

[1] Of the exponential family.

Consider again the typical linear regression. We assume a Gaussian (i.e. Normal) distribution for the response, we assume equal variance for all observations, and that there is a direct link of the linear predictor and the expected value μ, i.e. $X\beta = \mu$. In fact the typical linear regression model is a generalized linear model with a Gaussian distribution and identity link function.

$$g(\mu) = X\beta$$
$$g(\mu) = \eta$$
$$E(y) = \mu = g^{-1}(\eta)$$

To further illustrate the generalization, we consider a distribution other than the Gaussian. In the example to the right, we examine a Poisson distribution for some vector of count data. There is only one parameter to be considered, μ, since for the Poisson the mean and

$$y \sim \mathcal{P}(\mu)$$
$$g(\mu) = b_0 + b_1X_1 + b_2X_2 ... + b_pX_p$$

variance are equal. For the Poisson, the (canonical) link function $g(.)$, is the natural log, and so relates the log of μ to the linear predictor. As such we could also write it in terms of exponentiating the right hand side.

$$\mu = e^{b_0+b_1X_1+b_2X_2...+b_pX_p}$$

While there is a great deal to further explore with regard to generalized linear models, the point here is to simply to note them as a generalization of the typical linear model with which all would be familiar after an introductory course in statistics. As we eventually move to generalized additive models, we can see them as a subsequent step in the generalization.

Generalized Additive Model

Now let us make another generalization to incorporate nonlinear forms of the predictors. The form at the right gives the new setup relating our new, now nonlinear predictor to the expected value, with whatever link function may be appropriate.

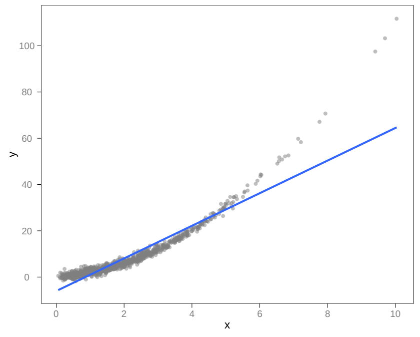
$$\begin{aligned} y &\sim ExpoFam(\mu, etc.) \\ \mu &= E(y) \\ g(\mu) &= b_0 + f(x_1) + f(x_2) + ... + f(x_p) \end{aligned}$$

So what's the difference? In short, we are using smooth functions of our predictor variables, which can take on a great many forms, with more detail in the following section. For now, it is enough to note the observed values y are assumed to be of some exponential family distribution, and μ is still related to the model predictors via a link function. The key difference now is that the linear predictor now incorporates smooth functions $f(x)$ of at least some (possibly all) covariates.

Beyond the General Linear Model II

Fitting the Standard Linear Model

In many data situations the relationship between some covariate(s) X and response y is not so straightforward. Consider the plot at the right. Attempting to fit a standard OLS regression results in the blue line, which doesn't capture the relationship very well.

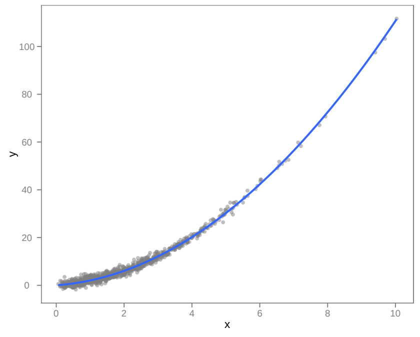


Polynomial Regression

One common approach we could undertake is to add a transformation of the predictor X, and in this case we might consider a quadratic term such that our model looks something like that to the right.

$$\begin{aligned} y &\sim \mathcal{N}(\mu, \sigma^2) \\ \mu &= b_0 + b_1X_1 + b_2X^2 \end{aligned}$$

We haven't really moved from the general linear model in this case, but we have a much better fit to the data as evidenced by the graph.

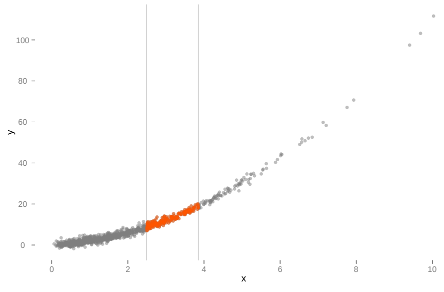


Scatterplot Smoothing

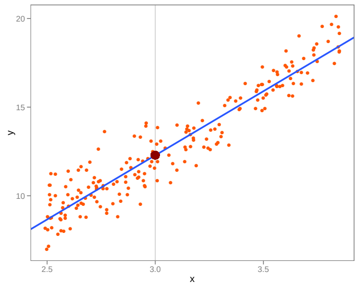
There are still other possible routes we could take. Many are probably familiar with *lowess* (or loess) regression, if only in the sense that often statistical packages may, either by default or with relative ease, add a nonparametric fit line to a scatterplot[2]. By default this is typically a lowess fit.

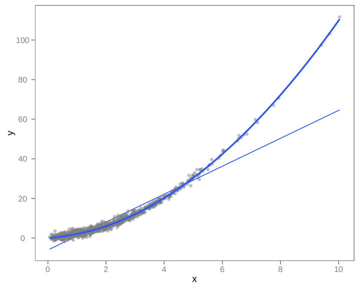
[2] See the scatterplots in the *car* package for example

Now look at the figure to the right. For every (sorted) x_0 value, we choose a neighborhood around it and, for example, fit a simple regression. As an example, let's look at $x_0 = 3.0$, and choose a neighborhood of 100 X values below and 100 values above. Now, for just that range we fit a linear regression model and note the fitted value where $x_0 = 3.0$. If we now do this for *every* X value, we'll have fitted values based on a rolling neighborhood that is relative to each X value being considered. Other options we could throw into this process, and usually do, would be to fit a polynomial regression for each neighborhood, and weight observations the closer they are to the X value, with less weight given to more distant observations.



The next plot shows the result from such a fitting process, specifically *lowess*, or *locally weighted scatterplot smoothing*. For comparison the regular regression fit is also provided. Even without using a lowess approach, we could fit have fit a model assuming a quadratic relationship, $y = x + x^2$, and it would result in a far better fit than the simpler model[3]. While polynomial regression might suffice in some cases, the issue is that nonlinear relationships are generally not specified so easily[4]. One can see this a variety of approaches in the bottom figure, which is a version of that found in Venables and Ripley (1997 edition). The leftmost plots are from a polynomial regression and lowess fit.



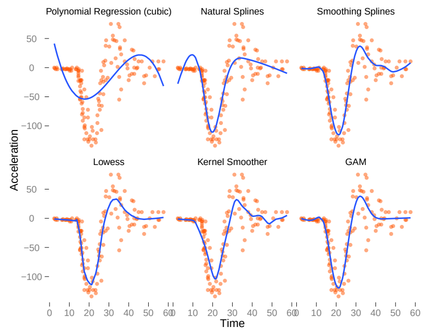


[3] y was in fact constructed as $x + x^2 +$ noise.

[4] Bolker 2008 notes an example of fitting a polynomial to 1970s U.S. Census data that would have predicted a population crash to zero by 2015.

Generalized Additive Models

The last figure on this page regards a data set giving a series of measurements of head acceleration in a simulated motorcycle accident. Time is in milliseconds, acceleration in g. Here we have data that are probably not going to be captured with simple transformations of the predictors. In the figure one can compare various approaches, and the first is a straightforward cubic polynomial regression, which unfortunately doesn't help us much. We could try higher orders, which would help, but in the end we will need something more flexible, and generalized additive models can help us in such cases.



Summary

Let's summarize our efforts up to this point.

Standard Linear Model
$y \sim \mathcal{N}(\mu, \sigma^2)$
$\mu = b_0 + b_1 X_1$

Polynomial Regression
$y \sim \mathcal{N}(\mu, \sigma^2)$
$\mu = b_0 + b_1 X_1 + b_2 X^2$

GLM formulation
$y \sim \mathcal{N}(\mu, \sigma^2)$
$g(\mu) = b_0 + b_1 X_1 + b_2 X^2$ identity link function

GAM formulation
$y \sim \mathcal{N}(\mu, \sigma^2)$
$g(\mu) = f(X)$ identity link function

Now that some of the mystery will hopefully have been removed, let's take a look at GAMs in action.

Application Using R

Initial Examination

THE DATA SET HAS BEEN CONSTRUCTED using average Science scores by country from the Programme for International Student Assessment (PISA) 2006, along with GNI per capita (Purchasing Power Parity, 2005 dollars), Educational Index, Health Index, and Human Development Index from UN data. The key variables are as follows (variable abbreviations in bold):

Overall Science Score (average score for 15 year olds)

Interest in science

Support for scientific inquiry

Income Index

Health Index

Education Index

Human **D**evelopment **I**ndex (composed of the Income index, Health Index, and Education Index)

The education component is measured by mean of years of schooling for adults aged 25 years and expected years of schooling for children of school entering age, the health index by life expectancy at birth, and the wealth component is based on the gross national income per capita. The HDI sets a minimum and a maximum for each dimension, and values indicate where each country stands in relation to these endpoints, expressed as a value between 0 and 1. More information on the HDI measures can be found here.

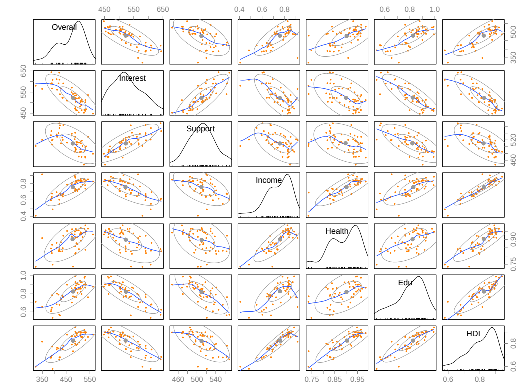
But even before we get too far, it would be good to know our options in the world of GAMs. Near the end of this document is a list of some packages to be aware of, and there is quite a bit of GAM functionality available within R, even for just plotting[5]. For our purposes we will use *mgcv*.

[5] *ggplot2* has basic gam functionality for scatterplot smoothing. Examples are further in the text.

The first thing to do is get the data in and do some initial inspections.

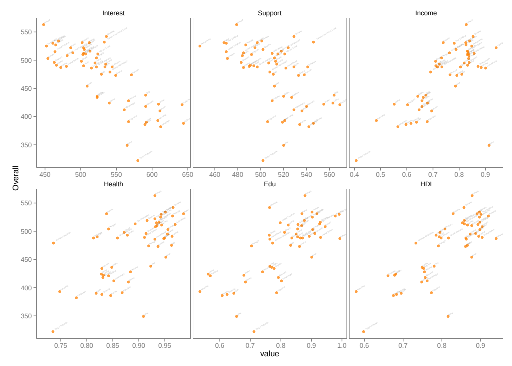
```
d = read.csv("http://www.nd.edu/~mclark19/learn/data/pisasci2006.csv")
library(psych)
describe(d)[-1, 1:9]  #univariate

##           var  n   mean    sd median trimmed   mad    min    max
## Overall     2 57 473.14 54.58 489.00  477.45 48.93 322.00 563.00
## Issues      3 57 469.91 53.93 489.00  474.28 41.51 321.00 555.00
## Explain     4 57 475.02 54.02 490.00  478.85 47.44 334.00 566.00
## Evidence    5 57 469.81 61.74 489.00  475.11 54.86 288.00 567.00
## Interest    6 57 528.16 49.84 522.00  525.72 53.37 448.00 644.00
## Support     7 57 512.18 26.08 512.00  512.15 25.20 447.00 569.00
## Income      8 61   0.74  0.11   0.76    0.75  0.11   0.41   0.94
## Health      9 61   0.89  0.07   0.89    0.89  0.08   0.72   0.99
## Edu        10 59   0.80  0.11   0.81    0.80  0.12   0.54   0.99
## HDI        11 59   0.81  0.09   0.82    0.81  0.09   0.58   0.94
```

```
library(car)
scatterplotMatrix(d[,-c(1,3:5)],pch=19,cex=.5,reg.line=F, lwd.smooth=1.25,
                  spread=F,ellipse=T, col=c('gray60','#2957FF','#FF8000'),
                  col.axis='gray50')
```



The scatterplot matrix has quite a bit of information to spend some time with- univariate and bivariate density, as well as loess curves. For our purposes we will ignore the issue regarding the haves vs. have-nots on the science scale and save that for another day. Note the surprising negative correlation between interest and overall score for science, which might remind us of Simpson's paradox, namely, that what occurs for the individual may not be the same for the group. One will also note that while there is a positive correlation between Income and Overall science scores, it flattens out after an initial rise. Linear fits might be difficult to swallow for human development subscales in general, but let's take a closer look[6].

```
library(ggplot2); library(reshape2)
#get data into a form to take advantage of ggplot
dmelt = melt(d, id=c('Country','Overall'),
             measure=c('Interest','Support','Income','Health','Edu','HDI'))

#leave the smooth off for now
ggplot(aes(x=value,y=Overall), data=dmelt) +
  geom_point(color='#FF8000',alpha=.75) +
  #geom_smooth(se=F) +
  geom_text(aes(label=Country), alpha=.25, size=1,angle=30, hjust=-.2,
            vjust=-.2) +
```



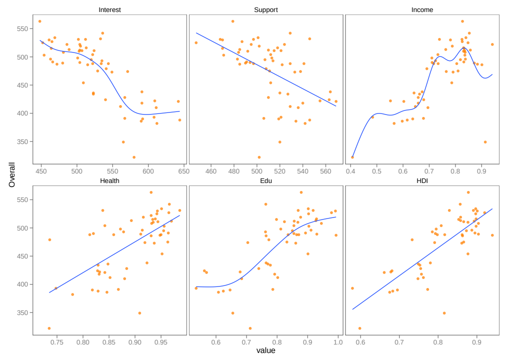
Zoom in to see country names.

[6] In the following plots I have a ggplot object called *ggtheme* to clean up the default plotting scheme. See the appendix.

```
  facet_wrap(~variable, scales='free_x') +
  ggtheme
```

We can see again that linear fits aren't going to do so well for some, though it might be a close approximation for interest in science and support for science. Now let's run the smooth. By default *ggplot2* will use a loess smoother for small data sets (i.e. $<$ 1000 observations), but can use the *mgcv* `gam` function as a smoother, though you'll need to load either the *gam* or *mgcv* packages first.

```
library(mgcv)
ggplot(aes(x=value,y=Overall), data=dmelt) +
  geom_point(color='#FF8000',alpha=.75) +
  geom_smooth(se=F, method='gam', formula=y~s(x), color='#2957FF') +
  facet_wrap(~variable, scales='free_x') +
  ggtheme
```



Often in these situations people will perform some standard transformation, such as a log, but it doesn't help as nearly as often as it is used. For example, in this case one can log the overall score, Income, or both and a linear relation will still not be seen.

Single Predictor

Linear Fit

We will start with the simple situation of a single predictor. Let's begin by using a typical linear regression to predict science scores by the Income index. We could use the standard R `lm` function, but I'll leave that as an exercise for comparison. We can still do straightforward linear models with the `gam` function, and again it is important to note that the standard linear model can be seen as a special case of a GAM.

```
library(mgcv)
mod_lm <- gam(Overall ~ Income, data = d)
summary(mod_lm)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ Income
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    204.3       35.4    5.78  4.3e-07 ***
## Income         355.9       46.8    7.61  5.4e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
```

```
## R-sq.(adj) =  0.518   Deviance explained = 52.7%
## GCV score = 1504.5  Scale est. = 1448.8    n = 54
```

What are we getting here? The same thing you get from a regular linear model, because you just ran one. However there are a couple things to look at. The coefficient is statistically significant, but serves as a reminder that it usually a good idea to scale predictor variables so that the effect is more meaningful. Here, moving one unit on Income is akin from going broke to being the richest country. But in a more meaningful sense, if we moved from say, .7 to .8, we'd expect an increase of about 35 points on the science score. We also see the deviance explained, which serves as a generalization of R-squared [7] , and in this case, it actually is equivalent to R-squared. Likewise there is the familiar adjusted version of it to account for small sample size and model complexity. The scale estimate is the scaled deviance, which here is equivalent to the residual sums of squares. The GCV score we will save for when we run a GAM.

[7] For those more familiar with generalized linear models, this is calculated as (Dev$_{Null}$-Dev$_{Residual}$)/Dev$_{Null}$
One can verify this by running the same model via the glm, and using the values from the summary of the model object.

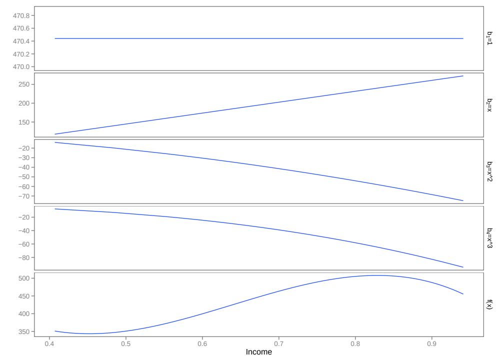
GAM

Let's now try some nonlinear approaches, keeping in mind that $\mu = f(x)$. As a point of comparison, we can start by trying a standard polynomial regression, and it might do well enough[8]. To start we must consider a basis for use, i.e. a set of basis functions b_j that will be combined to produce $f(x)$:

$$f(x) = \sum_{j=1}^{q} b_j(x)\beta_j$$

[8] The following example is pretty much straight from Wood (2006, Chapter 3) with little modification.

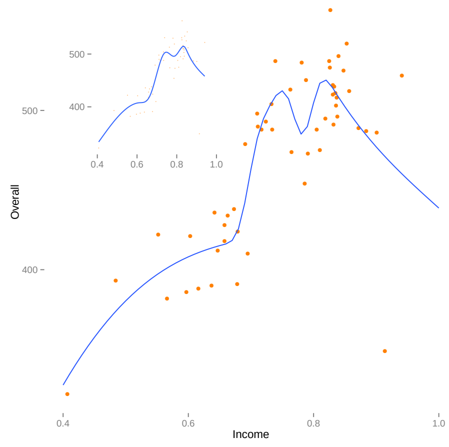
To better explain by example, if we had a cubic polynomial the basis is: $b_1(x) = 1$, $b_2(x) = x$, $b_3(x) = x^2$, $b_4(x) = x^3$, which, when combined give us our $f(x)$. For those familiar with the mechanics of linear modeling, this is just our 'model matrix', which we can save out with many model functions in R via the argument `x=T` or using the `model.matrix` function. With the graph at the right we can see the effects in action. It is based on the results extracted from running the model[9] and obtaining the coefficients (e.g. the first plot represents the intercept of 470.44, the second plot, our b_2 coefficient of 289.5 multiplied by Income and so forth). The bottom plot shows the final fit $f(x)$, i.e. the linear combination of the basis functions.



[9] see ?`poly` for how to fit a polynomial in R.

At this point we have done nothing we couldn't do in our regular regression approach, but the take home message is that as we move to GAMs we are going about things in much the same fashion; we are simply changing the nature of the basis, and have a great deal more flexibility in choosing the form.

In the next figure I show the fit using a 'by-hand' cubic spline basis (see the appendix and Wood, 2006, p.126-7). A cubic spline is essentially a connection of multiple cubic polynomial regressions. We choose points of the variable at which to create sections, and these points are referred to as *knots*. Separate cubic polynomials are fit at each section, and then joined at the knots to create a continuous curve. The graph represents a cubic spline with 8 knots[10] between the first and third quartiles. The inset graph uses the GAM functionality within ggplot2's `geom_smooth` as a point of comparison.



[10] Ten including the endpoints.

Let's now fit an actual generalized additive model using the same cubic spline as our smoothing function. We again use the `gam` function as before for basic model fitting, but now we are using a function `s` within the formula to denote the smooth terms. Within that function we also specify the type of smooth, though a default is available. I chose 'cr', denoting cubic regression splines, to keep consistent with our previous example.

```
mod_gam1 <- gam(Overall ~ s(Income, bs = "cr"), data = d)
summary(mod_gam1)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   470.44       4.08     115   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df    F p-value
## s(Income) 6.9   7.74 16.4   2e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =    0.7   Deviance explained = 73.9%
## GCV score = 1053.7  Scale est. = 899.67    n = 54
```

The first thing to note is, that aside from the smooth part, our model code is similar to what we're used to with core R functions such as lm and glm. In the summary, we first see the distribution assumed as well as the link function used, in this case normal and identity, respectively, which to iterate, had we had no smoothing would result in a standard regression model. After that we see that the output is separated into *parametric* and *smooth*, or nonparametric parts. In this case, the only parametric component is the intercept, but it's good to remember that you are not bound to smooth every effect of interest, and indeed, as we will discuss in more detail later, part of the process may involve refitting the model with terms that were found to be linear for the most part anyway. The smooth component of our model regarding a country's income's relationship with overall science score is statistically significant, but there are a couple of things in the model summary that would be unfamiliar.

We'll start with the *effective degrees of freedom*, or *edf*. In typical OLS regression the model degrees of freedom is equivalent to the number of predictors/terms in the model. This is not so straightforward with a GAM due to the smoothing process and the penalized regression estimation procedure, something that will be discussed more later[11]. In this situation, we are still trying to minimize the residual sums of squares, but also have a built in penalty for 'wiggliness' of the fit, where in general we try to strike a balance between an undersmoothed fit and an oversmoothed fit. The default p-value for the test is based on the effective degrees of freedom and the rank r of the covariance matrix for the coefficients for a particular smooth, so here *conceptually* it is the p-value associated with the $F(r, n - edf)$. However there are still other issues to be concerned about, and ?summary.gam will provide your first step down that particular rabbit hole. For hypothesis testing an alternate edf is actually used, which is the other one provided there in the summary result[12]. At this point you might be thinking these p-values are a bit fuzzy, and you'd be right. The gist is, they aren't to be used for harsh cutoffs, say, at an arbitrary .05 level[13], but if they are pretty low you can feel comfortable claiming statistical significance, which of course is the end all, be all, of the scientific endeavor.

[11] In this example there are actually 9 terms associated with this smooth, but they are each 'penalized' to some extent and thus the edf does not equal 9.

[12] Here it is noted 'Rel.df' but if, for example, the argument $p.type = 5$ is used, it will be labeled 'Est.Rank'. Also, there are *four* p-value types one can choose from. The full story of edf, p-values and related is scattered throughout Wood's text. See also ?`anova.gam`

[13] But then, standard p-values shouldn't be used that way either.

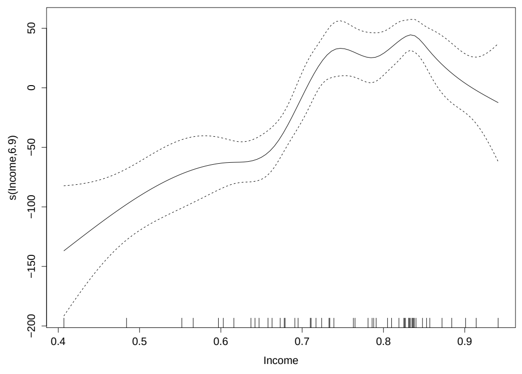
The GCV, or *generalized cross validation* score can be taken as an estimate of the *mean square prediction error* based on a leave-one-out cross validation estimation process. We estimate the model for all observations except i, then note the squared residual predicting observation i from the model. Then we do this for all observations. However, the GCV score is an efficient measure of this concept that doesn't actually require fitting all those models and overcomes other issues. It is this score that is minimized when determining the specific nature of the smooth. On its own it doesn't tell us much, but we can use it similar to AIC as a comparative measure to choose among different models, with lower being better.

In this initial model the GCV can be found as:
$GCV = \frac{n*scaled\,est.}{(n-edf-[n\,of\ parametric\,terms])^2}$

Graphical Display

One can get sense of the form of the fit by simply plotting the model object as follows:

```
plot(mod_gam1)
```



The intervals are Bayesian credible intervals.

Although in this single predictor case one can also revisit the previous graph, where the inset was constructed directly from ggplot. You can examine the code in the appendix.

Model Comparison

Let us now compare our regular regression fit to the GAM model fit. The following shows how one can extract various measures of performance and the subsequent table shows them gathered together.

```
AIC(mod_lm)

## [1] 550.2

summary(mod_lm)$sp.criterion

## [1] 1504

summary(mod_lm)$r.sq  #adjusted R squared

## [1] 0.5175
```

Do the same to extract those same elements from the GAM. The following display[14] makes for easy comparison.

[14] I just gathered the values into a data.frame object and used xtable from the eponymous package to produce the table.

aic_lm	aic_gam1	gcv_lm	gcv_gam1	rsq_lm	rsq_gam1
550.24	529.81	1504.50	1053.73	0.52	0.70

Comparing these various measures, it's safe to conclude that the GAM fits better. We can also perform the familiar statistical test via the anova function we apply to other R model objects. As with the previous p-value issue, we can't be too particular, and technically one could have a model with more terms but lower *edf*, where it just wouldn't even make sense[15]. As it would be best to be conservative, we'll proceed cautiously.

[15] ?anova.gam for more information.

```
anova(mod_lm, mod_gam1, test = "Chisq")

## Analysis of Deviance Table
##
## Model 1: Overall ~ Income
## Model 2: Overall ~ s(Income, bs = "cr")
##   Resid. Df Resid. Dev  Df Deviance Pr(>Chi)
## 1      52.0      75336
## 2      46.1      41479 5.9    33857  1.2e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It would appear the anova results tell us what we have probably come to believe already, that incorporating nonlinear effects has improved the model considerably.

Multiple Predictors

Let's now see what we can do with a more realistic case where we have added model complexity.

Linear Fit

We'll start with the typical linear model approach again, this time adding the Health and Education indices.

```
mod_lm2 <- gam(Overall ~ Income + Edu + Health, data = d)
summary(mod_lm2)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ Income + Edu + Health
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    121.2       79.0    1.53    0.131
## Income         182.3       85.3    2.14    0.038 *
## Edu            234.1       54.8    4.27  9.1e-05 ***
## Health          27.0      134.9    0.20    0.842
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.616   Deviance explained = 63.9%
## GCV score = 1212.3  Scale est. = 1119      n = 52
```

It appears we have statistical effects for Income and Education, but not for Health, and the adjusted R-squared suggests a notable amount of the variance is accounted for[16]. Let's see about nonlinear effects.

[16] Note that a difference in sample sizes do not make this directly comparable to the first model.

GAM

As far as the generalized additive model goes, we can approach things in a similar manner as before. We will ignore the results of the linear model for now and look for nonlinear effects for each covariate.

The default smoother for s() is the argument bs='tp', a thin plate regression spline.

```
mod_gam2 <- gam(Overall ~ s(Income) + s(Edu) + s(Health), data = d)
summary(mod_gam2)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income) + s(Edu) + s(Health)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   471.15       2.77     170   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df    F p-value
## s(Income) 7.59   8.41 8.83 1.3e-07 ***
## s(Edu)    6.20   7.18 3.31  0.0073 **
## s(Health) 1.00   1.00 2.74  0.1066
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.863   Deviance explained = 90.3%
## GCV score = 573.83  Scale est. = 399.5     n = 52
```

There are again a couple things to take note of. First, statistically speaking, we come to the same conclusion as the linear model regarding the individual effects. One should take particular note of the effect of Health index. The effective degrees of freedom with value 1 suggests that it has essentially been reduced to a simple linear effect. The following will update the model to explicitly model the effect as such, but as one can see, the results are identical.

```
mod_gam2B = update(mod_gam2, . ~ . - s(Health) + Health)
summary(mod_gam2B)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income) + s(Edu) + Health
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)      640        102    6.26  3.1e-07 ***
## Health          -190        115   -1.65     0.11
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df    F p-value
## s(Income) 7.59   8.41 8.83 1.3e-07 ***
## s(Edu)    6.20   7.18 3.31  0.0073 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.863   Deviance explained = 90.3%
## GCV score = 573.83  Scale est. = 399.5     n = 52
```

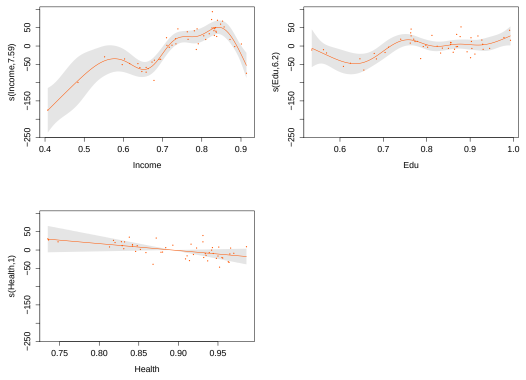
We can also note that this model accounts for much of the variance in Overall science scores, with an adjusted R-squared of .86. In short, it looks like the living standards and educational resources of a country are associated with overall science scores, even if we don't really need the Health index in the model.

Graphical Display

Now we examine the effects of interest visually. In the following code, aside from the basic plot we have changed the default options to put all the plots on one page, added the partial residuals[17], and changed the symbol and its respective size, among other options.

[17] See e.g. John Fox's texts and chapters regarding regression diagnostics, and his crplots function in the *car* package.

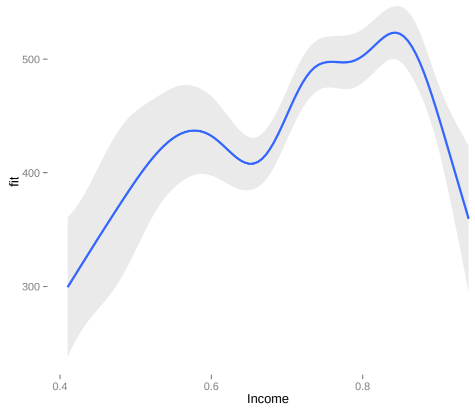
```
plot(mod_gam2, pages=1, residuals=T, pch=19, cex=0.25,
     scheme=1, col='#FF8000', shade=T,shade.col='gray90')
```



Here we can see the effects of interest, and again one might again note the penalized-to-linear effect of Health. We again see the tapering off of Income's effect at its highest level, and in addition, a kind of sweet spot for a positive effect of Education in the mid-range values, with a slight positive effect overall. Health, as noted, has been reduced to a linear, surprisingly negative effect, but again this is non-significant. One will also note the y-axis scale. The scale is on that of the linear predictor, but due to identifiability constraints, the smooths must sum to zero, and thus are presented in a mean-centered fashion.

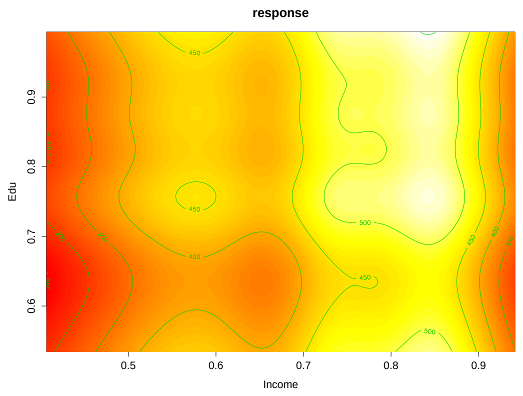
Previously, we had only one predictor and so we could let ggplot do the work to produce a plot on the response scale, although we could have added the intercept back in. Here we'll need to do a little more. The following will produce a plot for Income on the scale of the response, with the other predictors held at their mean. Seen at right.

```
# Note that mod_gam2$model is the data that was used in the modeling process,
# so it will have NAs removed.
testdata = data.frame(Income=seq(.4,1, length=100),
                      Edu=mean(mod_gam2$model$Edu),
                      Health=mean(mod_gam2$model$Health))
fits = predict(mod_gam2, newdata=testdata, type='response', se=T)
predicts = data.frame(testdata, fits)

ggplot(aes(x=Income,y=fit), data=predicts) +
  geom_smooth(aes(ymin = fit - 1.96*se.fit, ymax=fit + 1.96*se.fit),
              fill='gray80', size=1,stat='identity') +
  ggtheme
```



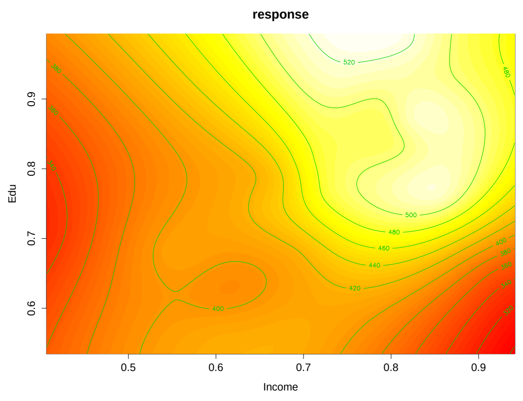
This gives us a sense for one predictor, but let's take a gander at Income and Education at the same time. Previously, while using the function `plot`, it actually is `plot.gam`, or the basic plot function for a GAM class object. There is another plotting function, `vis.gam`, that will give us a bit more to play with. The following will produce a contour plot (directly to the right) with Income on the x axis, Education on the y, with values on the response scale given by the contours, with lighter color indicating higher values.

```
vis.gam(mod_gam2, type = "response", plot.type = "contour")
```

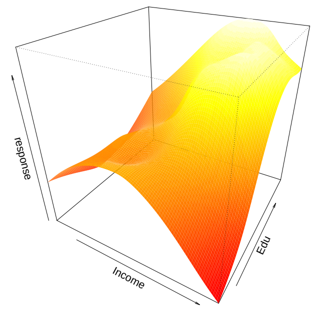


First and foremost the figure to the right reflects the individual plots (e.g. one can see the decrease in scores at the highest income levels), and we can see that middling on Education and high on Income generally produces the highest scores. Conversely, being low on both the Education and Income indices are associated with poor Overall science scores. However, while interesting, these respective smooths were created separately of one another, and there is another way we might examine how these two effects work together in predicting the response.

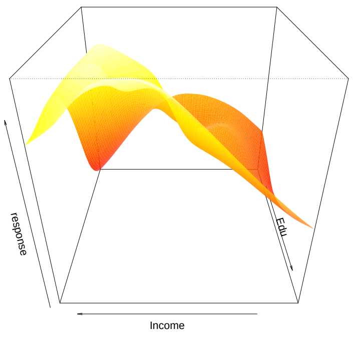
Let's take a look at another approach, continuing the focus on visual display. It may not be obvious at all, but one can utilize smooths of more than one variable, in effect, a smooth of the smooths of the variables that go into it. Let's create a new model to play around with this feature. After fitting the model, an example of the plot code is given producing the middle figure, but I also provide a different perspective as well as contour plot for comparison with the graph based on separate smooths.



```
mod_gam3 <- gam(Overall ~ te(Income, Edu), data = d)
summary(mod_gam3)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ te(Income, Edu)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   471.15       3.35     141   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                  edf Ref.df    F p-value
## te(Income,Edu) 10.1   12.2 16.9  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =    0.8   Deviance explained =   84%
## GCV score = 741.42  Scale est. = 583.17    n = 52
```



```
vis.gam(mod_gam3, type='response', plot.type='persp',
        phi=30, theta=30,n.grid=500, border=NA)
```



In the above we are using a type of smooth called a tensor product smooth, and by smoothing the marginal smooths of Income and Education, we see a bit clearer story. As we might suspect, wealthy countries with more of an apparent educational infrastructure are going to score higher on the Overall science score. However, wealth is not necessarily indicative of higher science scores (note the dark bottom right corner on the contour plot)[18], though without at least moderate wealth hopes

[18] This is likely due to Qatar. Refer again to the figure on page 9.

are fairly dim for a decent score.

One can also, for example, examine interactions between a smooth and a linear term $f(x)z$, and in a similar vein of thought look at smooths at different levels of a grouping factor. We'll leave that for some other time.

Model Comparison

As before, we can examine indices such as GCV or perhaps adjusted R-square, which both suggest our GAM performs considerably better. Statistically we can compare the two models with the anova function as before.

```
anova(mod_lm2, mod_gam2, test = "Chisq")

## Analysis of Deviance Table
##
## Model 1: Overall ~ Income + Edu + Health
## Model 2: Overall ~ s(Income) + s(Edu) + s(Health)
##   Resid. Df Resid. Dev   Df Deviance Pr(>Chi)
## 1      48.0      53713
## 2      36.2      14463 11.8    39250  9.8e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Not that we couldn't have assumed as such already, but now we have additional statistical evidence to suggest that incorporating non-linear effects improves the model.

Other Issues

Estimation

As noted previously, estimation of GAMs in the mgcv package is conducted via a penalized likelihood approach. Conceptually this amounts to fitting the following model

$$g(\mu) = X\beta + f(x_1) + f(x_2)...f(x_p)$$

But note that each smooth has its own model matrix made up of the bases. So for each smooth covariate we have:

$$f_j = \tilde{X}_j\tilde{\beta}_j$$

Given a matrix of coefficients $\mathbf{S}$, we can more formally note a penalized likelihood function:

$$l_p(\beta) = l(\beta) - \frac{1}{2}\sum_j \lambda_j \beta^{\mathrm{T}} S_j \beta$$

where $l(\beta)$ is the usual GLM likelihood function, and λ_j are the smoothing parameters. The part of the function including λ penalizes curvature in the function, where λ establishes a trade-off between the goodness of fit and the smoothness, and such an approach will allow for less overfitting[19]. Technically we could specify the smoothing parameters explicitly, and the appendix has some 'by-hand' code taken directly from Wood (2006) with only slight modifications, where the smoothing parameters are chosen and compared[20].

[19] $\lambda \to \infty$ would result in a straight line estimate for f_j, while $\lambda = 0$ would allow any f that interpolates the data (Wood, 2006, p. 128)

[20] And we would have to choose in order for the penalized maximum likelihood to work.

Smoothing parameters however are in fact estimated, and this brings us back to the cross-validation procedure mentioned before. Smoothing parameters are selected which minimize the GCV score by default, though one has other options. Note that there are other approaches to estimation such as backfitting, generalized smoothing splines and Bayesian.

Shrinkage & Variable Selection

Some smooths are such that no matter the smoothing parameter, there will always be non-zero coefficients. An extra penalty may be added such that if the smoothing parameter is large enough, the coefficients will shrink to zero, and some smoothing bases will have such alternative approaches available[21]. In this manner one can assess whether a predictor is adding anything to the model, i.e. if it's effective degrees of freedom is near zero, and perhaps use the approach as a variable selection technique.

[21] See also, the *select* argument to the `gam` function. Also, in version 1.7-19, which came out during creation of this document, `summary.gam` and `anova.gam` had changes made to improve the p-value computation in penalized situations.

Effective degrees of freedom again

If we define a matrix **F** that maps the unpenalized estimates of β to the penalized estimates such that

$$F = (X^T X + S)^{-1} X^T X$$

and note

$$\tilde{\beta} = (X^T X)^{-1} X^T y$$
$$\hat{\beta} = F\tilde{\beta}$$

the diagonal elements of **F** are where the effective degrees of freedom for each covariate come from.

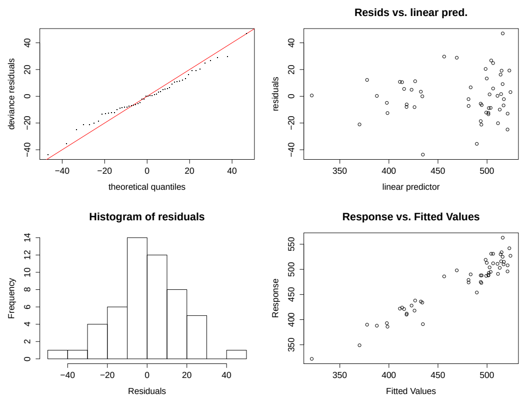
Choice of Smoothing Function

A number of smooths are available with the *mgcv* package, and one can learn more by entering ?`smooth.terms` at the command line. In our models we have used cubic regression splines and thin plate regression splines (TPRS), the latter being the default for a GAM in this package. As a brief summary, TPRS work well in general in terms of performance and otherwise has some particular advantages, and has a shrinkage alternative available. One should still feel free to play around, particularly when dealing with multiple smooths, where the tensor product smooths would be better for covariates of different scales.

Diagnostics & Choice of Basis Dimension

We have some built-in abilities to examine whether there are any particular issues, and we can try it with our second GAM model.

```
gam.check(mod_gam2, k.rep = 1000)

##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 21 iterations.
## The RMS GCV score gradiant at convergence was 2.499e-05 .
## The Hessian was positive definite.
## The estimated model rank was 28 (maximum possible: 28)
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##              k'   edf k-index p-value
## s(Income) 9.000 7.593   1.258    0.97
## s(Edu)    9.000 6.204   1.008    0.46
## s(Health) 9.000 1.000   0.899    0.18
```



The plots are of the sort we're used to from a typical regression setting, though it's perhaps a bit difficult to make any grand conclusion based on such a small data set. The printed output on the other hand contains unfamiliar information, but is largely concerned with over-smoothing, and so has tests of whether the basis dimension for a smooth is too low. The p-values are based on simulation, so I bumped up the number with the additional argument. Guidelines are given in the output itself, and at least in this case it does not look like we have an issue. However, if there were a potential problem, it is suggested to double k[22] and refit, and if the effective degrees of freedom increases quite a bit you would probably want to go with the updated model[23]. Given the penalization process, the exact choice of k isn't too big of a deal, but the defaults are arbitrary. You want to set it large enough to get at the true effect as best as possible, but in some cases computational efficiency will also be of concern. The help on the function `choose.k` provides another approach to examining k based on the residuals from the model under consideration, and provides other useful information.

One can inspect the quantile-quantile plot directly with the `qq.gam` function.

[22] k can be set as an argument to `s(var1, k=?)`.

[23] I actually did this for Health, which is the only one of the predictors that could be questioned, and there was no change at all; it still reduced to a linear effect.

Prediction

A previous example used the predict function on the data used to fit the model to obtain fitted values on the response scale. Typically we'd use this on new data. I do not cover it, because the functionality is the same as the `predict.glm` function in base R, and one can just refer to that. It is worth noting that there is an option, `type='lpmatrix'`, which will return the actual model matrix by which the coefficients must be pre-multiplied to get the values of the linear predictor at the supplied covariate values. This can be particularly useful towards opening the black box as one learns the technique.

Model Comparison Revisited

We have talked about automated smoothing parameter and term selection, and in general potential models are selected based on estimation of the smoothing parameter. Using an extra penalty to allow coefficients to tend toward zero with the argument `select=TRUE` is an automatic way to go about it, where some terms could effectively drop out. Otherwise we could compare models GCV/AIC scores[24], and in general either of these would be viable approaches. Consider the following comparison:

[24] GCV scores are not useful for comparing fits of different families; AIC is still ok though.

```
mod_1d = gam(Overall ~ s(Income) + s(Edu), data = d)
mod_2d = gam(Overall ~ te(Income, Edu, bs = "tp"), data = d)
AIC(mod_1d, mod_2d)

##           df   AIC
## mod_1d 15.59 476.1
## mod_2d 13.25 489.7
```

In some cases we might prefer to be explicit in comparing models with and without particular terms, and we can go about comparing models as we would with a typical GLM analysis of deviance. We have demonstrated this previously using the `anova.gam` function, where we compared linear fits to a model with an additional smooth function. While we could construct a scenario that is identical to the GLM situation for a statistical comparison, it should be noted that in the usual situation the test is actually an approximation, though it should be close enough when it is appropriate in the first place. The following provides an example that would nest the main effects of Income and Education within the product smooth, i.e. sets their basis dimension and smoothing function to the defaults employed by the te smooth.

```
mod_A = gam(Overall ~ s(Income, bs = "cr", k = 5) + s(Edu, bs = "cr", k = 5),
    data = d)
mod_B = gam(Overall ~ s(Income, bs = "cr", k = 5) + s(Edu, bs = "cr", k = 5) +
    te(Income, Edu), data = d)
```

```
anova(mod_A, mod_B, test = "Chi")

## Analysis of Deviance Table
##
## Model 1: Overall ~ s(Income, bs = "cr", k = 5) + s(Edu, bs = "cr", k = 5)
## Model 2: Overall ~ s(Income, bs = "cr", k = 5) + s(Edu, bs = "cr", k = 5) +
##     te(Income, Edu)
##   Resid. Df Resid. Dev   Df Deviance Pr(>Chi)
## 1      46.3      36644
## 2      42.0      24964 4.26    11679  0.00075 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Again though, we could have just used the summary output from the second model.

Instances where such an approach does not appear to be appropriate within the context of the *mgcv* package are when terms are able to be penalized to zero; in such a case p-values will be much too low. In addition, when comparing GAMs, sometimes the nesting of models would not be so clear when there are multiple smooths involved, and additional steps may need to be taken to make sure they are nested. We must make sure that each smooth term in the null model has no more effective degrees of freedom than the same term in the alternative, otherwise it's possible that the model with more terms can have lower effective degrees of freedom but better fit, rendering the test nonsensical. Wood (2006) suggests that if such model comparison is the ultimate goal an unpenalized approach[25] would be best to have much confidence in the p-values.[26]

[25] This can be achieved with the argument `s(..., fx=T)`, although now one has to worry more about the k value used, as very high k will lead to low power

[26] One might also examine the *gss* package for anova based approach to generalized smoothing splines.

Other Approaches

THIS SECTION WILL DISCUSS some ways to relate the generalized models above to other forms of nonlinear modeling approaches, some familiar and others perhaps less so. In addition, I will note some extensions to GAMs to consider.

Relation to Other Nonlinear Modeling Approaches

Known Form

It should be noted that one can place generalized additive models under a general heading of *nonlinear models* whose focus may be on transformations of the outcome (as with generalized linear models), the predictor variables (polynomial regression and GAMs), or both (GAMs), in addition to those whose effects are nonlinear in the parameters [27]. The difference between the current presentation and those

A general form of nonlinear models: $y = f(X, \beta) + \epsilon$

[27] For example, various theoretically motivated models in economics and ecology.

latter nonlinear models as distinguished in typical introductory statistical texts that might cover some nonlinear modeling, is that we simply don't know the form beforehand.

In cases where the form may be known, one can use an approach such as nonlinear least squares, and there is inherent functionality within a standard R installation, such as the `nls` function. As is the usual case, such functionality is readily extendable to a great many other analytic situations, e.g. the *gnm* for generalized nonlinear models or *nlme* for nonlinear mixed effects models.

Response Transformation

It is common practice, perhaps too common, to manually transform the response and go about things with a typical linear model. While there might be specific reasons for doing so, the primary reason applied researchers seem to do so is to make the distribution 'more normal' so that regular regression methods can be applied. As an example, a typical transformation is to take the log, particularly to tame 'outliers' or deal with skewness.

While it was a convenience 'back in the day' because we didn't have software or computing power to deal with a lot of data situations aptly, this is definitely not the case now. In many situations it would be better to, for example, conduct a generalized linear model with a log link or perhaps assume a different distribution for the response directly (e.g. skew-normal), and many tools allow researchers to do this with ease[28].

[28] A lot of 'outliers' tend to magically go away with an appropriate choice of distribution for the data generating process.

There are still cases where one might focus on response transformation, just not so one can overcome some particular nuisance in trying to fit a linear regression. An example might be in some forms of *functional data analysis*, where we are concerned with some function of the response that has been measured on many occasions over time.

The Black Box

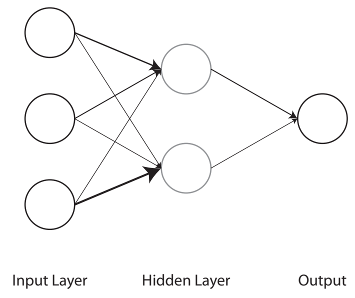
Venables and Ripley (2002, Section 11.5) make an interesting classification of nonlinear models into those that are less flexible but under full user control (fully parametric), and those that are *black box* techniques that are highly flexible and fully automatic: stuff goes in, stuff comes out, but we're not privy to the specifics[29].

One could probably make the case that most modeling is 'black box' for a great many researchers.

Two examples of the latter that they provide are *projection pursuit* and *neural net* models, though a great many would fall into such a heading. Projection pursuit models are well suited to high dimensional data where dimension reduction is a concern. One may think of an example where one uses a technique such as principal components analysis on the predictor set and then examines smooth functions of M principal components.

[29] For an excellent discussion of these different approaches to understanding data see Breiman (2001) and associated commentary. For some general packages outside of R that incorporate a purely algorithmic approach to modeling, you might check out RapidMiner or Weka. Note also that RapidMiner allows for R functionality via an add-on package, and the RWeka package brings Weka to the R environment.

In the case of neural net models, one can imagine a model where the input units (predictor variables) are weighted and summed to create hidden layer units, which are then essentially put through the same process to create outputs (see a simple example to the right). One can see projection pursuit models as an example where a smooth function is taken of the components which make up the hidden layer. Neural networks are highly flexible in that there can be any number of inputs, hidden layers, and outputs. However, such models are very explicit in the black box approach.



A Neural Net Model

Projection pursuit and neural net models are usually found among data mining/machine learning techniques any number of which might be utilized in a number of disciplines. Other more algorithmic/black box approaches include *k-nearest-neighbors*, *random forests*, *support vector machines*, and various tweaks or variations thereof including boosting, bagging, bragging and other alliterative shenanigans[30]. As Venables and Ripley note, generalized additive models might be thought of as falling somewhere in between the fully parametric and interpretable models of linear regression and black box techniques. Indeed, there are algorithmic approaches which utilize GAMs as part of their approach.

[30] See Hastie et al. (2009) for an overview of such approaches.

Extensions

Other GAMs

Note that just as generalized additive models are an extension of the generalized linear model, there are generalizations of the basic GAM beyond the settings described. In particular, random effects can be dealt with in this context as they can with linear and generalized linear models, and there is an interesting connection between smooths and random effects in general [31]. Generalized additive models for location, scale, and shape (GAMLSS) allow for distributions beyond the exponential family Rigby and Stasinopoulos (2005). In addition there are boosted, ensemble and other machine learning approaches that apply GAMs as well[32]. In short, there's plenty to continue to explore once one gets the hang of generalized additive models.

[31] Wood (2006) has a whole chapter devoted to the subject.

[32] See the *GAMens* package for example.

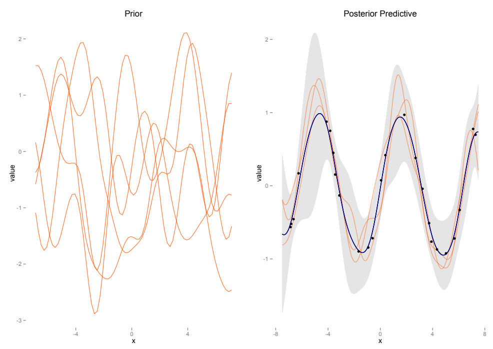
Gaussian Processes: a Probabilistic Approach

We can also approach modeling by using generalizations of the Gaussian distribution. Where the Gaussian distribution is over vectors and defined by a mean vector and covariance matrix, the *Gaussian Process* is over functions. A function f is distributed as a Gaussian Process defined by a mean function m and covariance function k.

$$f \sim \mathcal{GP}(m, k)$$

In the Bayesian context we can define a prior distribution over functions and make draws from a posterior predictive distribution of f. The reader is encouraged to consult Rasmussen and Williams (2006) for the necessary detail. The text is free for download here, and Rasmussen provides a nice and brief intro here. I also have some R code for demonstration here based on his Matlab code.

Suffice it to say in this context, it turns out that generalized additive models with a tensor product or cubic spline smooth are maximum a posteriori (MAP) estimates of gaussian processes with specific covariance functions and a zero mean function. In that sense one might segue nicely to those if familiar with additive models.



Gaussian Process $y = \sin(x) + \text{noise}$. The left graph shows functions from the prior distribution, the right shows the posterior mean function, 95% confidence interval shaded, as well as specific draws from the posterior predictive mean distribution.

Conclusion

Generalized additive models are a conceptually straightforward tool that allows one to incorporate nonlinear predictor effects into their otherwise linear models. In addition, they allow one to keep within the linear and generalized linear frameworks with which one is already familiar, while providing new avenues of model exploration and possibly improved results. As was demonstrated, it is easy enough with just a modicum of familiarity to pull them off within the R environment, and as such it is hoped that this document provides a means to do so for the uninitiated.

Appendix

R packages

THE FOLLOWING IS a non-exhaustive list of R packages which contain GAM functionality. Each is linked to the CRAN page for the package. Note also that several build upon the *mgcv* package used for this document.

amer: Fitting generalized additive mixed models based on the mixed model algorithm of lme4 (gamm4 now includes this approach).

CausalGAM: This package implements various estimators for average treatment effects.

COZIGAM: Constrained and Unconstrained Zero-Inflated Generalized Additive Models.

CoxBoost: This package provides routines for fitting Cox models. See also `cph` in rms package for nonlinear approaches in the survival context.

gam: Functions for fitting and working with generalized additive models.

GAMBoost: This package provides routines for fitting generalized linear and and generalized additive models by likelihood based boosting.

gamboostLSS: Boosting models for fitting generalized additive models for location, shape and scale (gamLSS models).

GAMens: This package implements the GAMbag, GAMrsm and GAMens ensemble classifiers for binary classification.

gamlss: Generalized additive models for location, shape, and scale.

gamm4: Fit generalized additive mixed models via a version of mgcv's gamm function.

gammSlice: Bayesian fitting and inference for generalized additive mixed models.

GMMBoost: Likelihood-based Boosting for Generalized mixed models.

gss: A comprehensive package for structural multivariate function estimation using smoothing splines.

mboost: Model-Based Boosting.

mgcv: Routines for GAMs and other generalized ridge regression with multiple smoothing parameter selection by GCV, REML or UBRE/AIC. Also GAMMs.

VGAM: Vector generalized linear and additive models, and associated models.

Miscellaneous Code

Here is any code that might be okay to play around with but would have unnecessarily cluttered sections of the paper.

ggtheme

My ggplot2 default options: ggtheme. Note that by actually adding the theme line after the ggtheme section, you can then override or change certain parts of it on the fly while maintaining the gist. I have this in my Rprofile.site file so that it is always available.

```
#set default options in case we use ggplot later
ggtheme =
  theme(
    axis.text.x = element_text(colour='gray50'),
    axis.text.y = element_text(colour='gray50'),
    panel.background = element_blank(),
    panel.grid.minor = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_rect(colour='gray50'),
    strip.background = element_blank()
  )
```

mod_gam1 plot

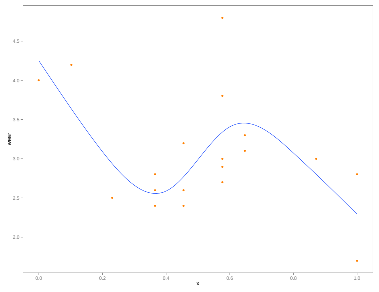
```
ggplot(aes(x=Income, y=Overall), data=d) +
  geom_point(color="#FF8000") +
  geom_smooth(se=F, method='gam', formula=y~s(x, bs="cr")) +
  xlim(.4,1) +
  ggtheme
```

Penalized Estimation Example

Initial data set up and functions.

```
############################
### Wood by-hand example ###
############################


size = c(1.42,1.58,1.78,1.99,1.99,1.99,2.13,2.13,2.13,
         2.32,2.32,2.32,2.32,2.32,2.43,2.43,2.78,2.98,2.98)
wear = c(4.0,4.2,2.5,2.6,2.8,2.4,3.2,2.4,2.6,4.8,2.9,
         3.8,3.0,2.7,3.1,3.3,3.0,2.8,1.7)
x= size-min(size); x = x/max(x)
d = data.frame(wear, x)

#cubic spline function
rk <- function(x,z) {
  ((z-0.5)^2 - 1/12)*((x-0.5)^2 - 1/12)/4-
    ((abs(x-z)-0.5)^4-(abs(x-z)-0.5)^2/2 + 7/240) / 24
}

spl.X <- function(x,knots){
  q <- length(knots) + 2  # number of parameters
  n <- length(x)          # number of observations
  X <- matrix(1,n,q)      # initialized model matrix
  X[,2] <- x              # set second column to x
  X[,3:q] <- outer(x,knots,FUN=rk) # remaining to cubic spline
  X
}

spl.S <- function(knots) {
  q = length(knots) + 2
  S = matrix(0,q,q)     # initialize matrix
  S[3:q,3:q] = outer(knots,knots,FUN=rk) # fill in non-zero part
  S
}

#matrix square root function
mat.sqrt <- function(S){
  d = eigen(S, symmetric=T)
  rS = d$vectors%*%diag(d$values^.5)%*%t(d$vectors)
  rS
}

#the fitting function
prs.fit <- function(y,x,knots,lambda){
  q = length(knots) + 2   # dimension of basis
  n = length(x)           # number of observations
  Xa = rbind(spl.X(x,knots), mat.sqrt(spl.S(knots))*sqrt(lambda))  # augmented model matrix
  y[(n+1):(n+q)] = 0   #augment the data vector
  lm(y ~ Xa-1)   # fit and return penalized regression spline
}
```

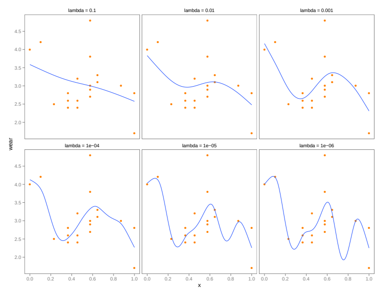
Example 1.



```
knots = 1:4/5
X = spl.X(x,knots)     # generate model matrix
mod.1 = lm(wear~X-1)   # fit model
xp <- 0:100/100        # x values for prediction
Xp <- spl.X(xp, knots) # prediction matrix

# Base R plot
plot(x, wear, xlab='Scaled Engine size', ylab='Wear Index', pch=19,
     col="#FF8000", cex=.75, col.axis='gray50')
lines(xp,Xp%*%coef(mod.1), col='#2957FF')   #plot

# ggplot
library(ggplot2)
ggplot(aes(x=x, y=wear), data=data.frame(x,wear))+
  geom_point(color="#FF8000") +
  geom_line(aes(x=xp, y=Xp%*%coef(mod.1)), data=data.frame(xp,Xp), color="#2957FF") +
  ggtheme
```

Example 2.

```
knots = 1:7/8

d2 = data.frame(x=xp)

for (i in c(.1,.01,.001,.0001,.00001,.000001)){
  mod.2 = prs.fit(wear, x, knots,i)   #fit penalized regression
                                      #spline choosing lambda
  Xp = spl.X(xp,knots) #matrix to map parameters to fitted values at xp
  d2[,paste('lambda = ',i, sep="")] = Xp%*%coef(mod.2)
}

### ggplot
library(ggplot2); library(reshape)
d3 = melt(d2, id='x')

ggplot(aes(x=x, y=wear), data=d) +
  geom_point(col='#FF8000') +
  geom_line(aes(x=x,y=value), col="#2957FF", data=d3) +
  facet_wrap(~variable) +
  ggtheme

### Base R approach
par(mfrow=c(2,3))
for (i in c(.1,.01,.001,.0001,.00001,.000001)){
  mod.2 = prs.fit(wear, x, knots,i)
  Xp = spl.X(xp,knots)
  plot(x,wear, main=paste('lambda = ',i), pch=19,
       col="#FF8000", cex=.75, col.axis='gray50')
  lines(xp,Xp%*%coef(mod.2), col='#2957FF')
}
```



R Session Info

R version 3.0.2 (2013-09-25)

Base packages: base, datasets, graphics, grDevices, grid, methods, stats, utils

Other packages: psych:1.3.10.12 mgcv:1.7-27 nlme:3.1-113 gridExtra:0.9.1 reshape2:1.2.2 ggplot2:0.9.3.1 MASS:7.3-29

References

Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231.

Fox, J. (2000a). *Multiple and Generalized Nonparametric Regression*. SAGE.

Fox, J. (2000b). *Nonparametric Simple Regression: Smoothing Scatterplots*. SAGE.

Hardin, J. W. and Hilbe, J. M. (2012). *Generalized Linear Models and Extensions, Third Edition*. Stata Press.

Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*. CRC Press.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer, 2nd ed. 2009. corr. 3rd printing 5th printing. edition.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press, Cambridge, Mass.

Rigby, R. A. and Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3).

Ruppert, D., Wand, M. P., and Carroll, R. J. (2003). *Semiparametric Regression*. Cambridge University Press.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics With S*. Birkhäuser.

Wood, S. N. (2006). *Generalized additive models: an introduction with R*, volume 66. CRC Press.