

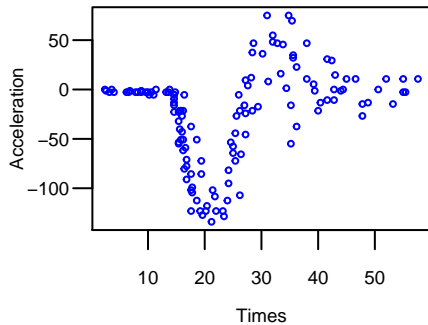
STATS 330

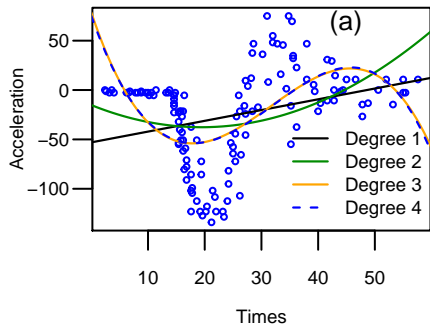
Handout 8

Generalised Additive Models

Department of Statistics, University of Auckland

A plot



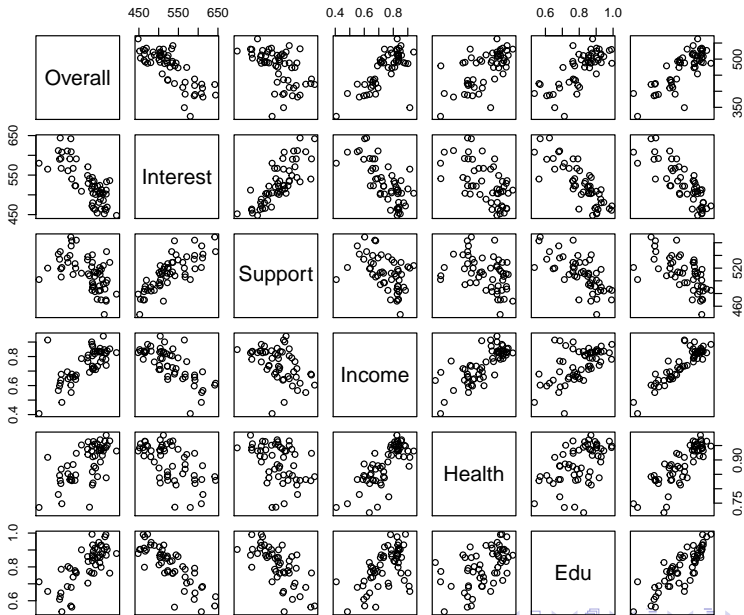


Motivation - PISA Example

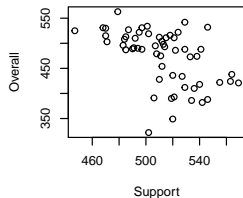
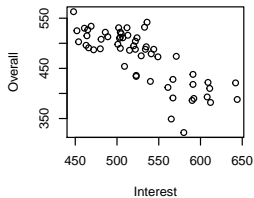
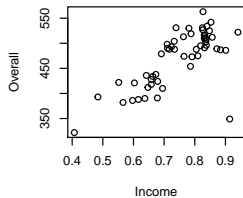
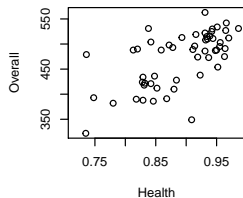
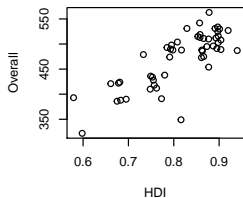
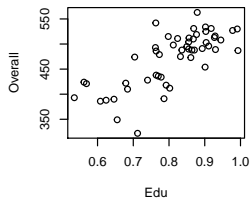
The `pisa2006` data set consists of average science scores by country from the Programme for International Student Assessment (PISA) 2006, with the following variables:

- ▶ **O**verall Science Score (average score for 15 year olds)
- ▶ **I**nterest in science
- ▶ **S**upport for scientific inquiry
- ▶ **I**ncome Index
- ▶ **H**ealth Index
- ▶ **E**ducation Index
- ▶ **H**uman **D**evelopment **I**ndex (composed of the Income, Health and Education Indices)

PISA Example



PISA Example



General Ideas

- ▶ The purpose of this chapter is to cover the basic ideas of *generalized additive models* (GAMs). This topic is potentially very theoretical, however we want to use GAMs practically and competently.
- ▶ Several **R** packages implement GAMs, and we will use a few of them. Each have their own flavour. Don't load more than one of them at any one time because `s()` is used by them all and they interfere with each other. Use `detach()` after their use.

General Ideas

- ▶ Much of the theory has been omitted and daggered (†) parts are non-examinable.
- ▶ Concentrate on the main *ideas* and *concepts* before focussing on the smaller details. The concepts are simple and should be straightforward to understand. Try to see the forest, not the trees!
- ▶ Some reference books will be mentioned along the way. . . they are supplementary reading only for those who want to pursue the topic later.
- ▶ Generalized additive modelling is an art as well as a science.

Introduction I

Recall, from Handout 1, that for GLMs we have the following:

$$g(\theta_i) = \beta_0 + \beta_1 x_i + \dots \quad (1)$$

where $Y_i \sim \text{Distribution}(\theta_i, \gamma)$ and g is an appropriate link function such as identity, log, logit. The effect of each X_i on $g(\theta_i)$ is *linear*. For additive models we relax the linearity assumption, using smoothers. Thus:

$$g(\theta_i) = \beta_0 + f_1(x_i) + \dots \quad (2)$$

where f_1 is a *smooth* function. GLMs are a special case of GAMs. So GLMs are a subclass of GAMs just as the LM is a special case of a GLM.

Introduction II

Why fit additive models?

- ▶ Since the functions are determined using smoothers, GAMs allow the data to determine the relationship between each explanatory variable and $g(\theta_i)$.
- ▶ Additivity is still assumed so GAMs are easy to interpret.
- ▶ It may suggest transformations of the explanatory variables that enable a GLM to be fitted instead.
- ▶ It helps confirm that a GLM is okay with respect to terms of the form $\beta_k x_k$.

Some GAM books: Harezlak, Ruppert and Wand (2018), Hastie and Tibshirani (1990), Ruppert, Wand and Carroll (2003), Wood (2006, 2017), Yee (2015). Perperoglou et al. (2019) describes several commonly-used spline-based **R** packages.

Smoothing

Smoothing is a powerful tool for exploratory data analysis (EDA).

Consider scatterplot data (x_i, y_i) , $i = 1, \dots, n$ where

$$y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim (0, \sigma) \quad (3)$$

Here, f is an arbitrary smooth function. The idea is to use a small window of data about x_0 to estimate $f(x_0)$. This approach is called **smoothing**.

Global models vs local models

GLMs are “global models”:

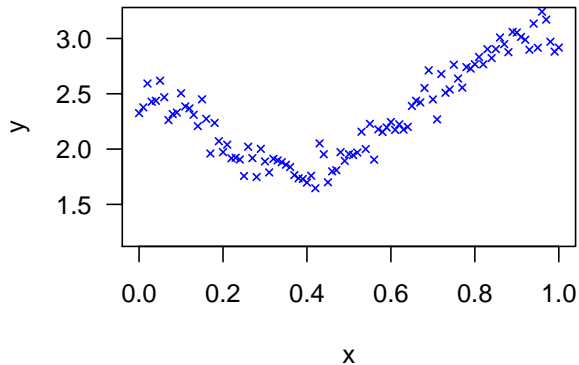
- ▶ the relationship $g(\theta_i) = \beta_0 + \beta_1 x_i + \dots$ is assumed to apply over the entire range of the explanatory variables,
- ▶ the fitted surface is determined by all the observations thus the fitted surface at any point $\hat{f}(x_0)$ depends on all the data,
- ▶ global models work well provided that the assumed model is not too complicated and provides a good approximation of the true relationship over the entire range of the explanatory variables.

Global models vs local models

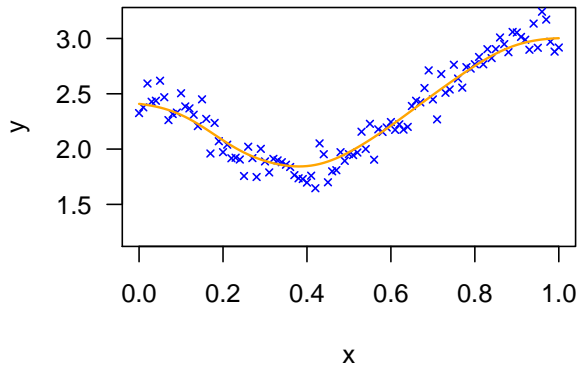
Local models (smoothers) only use data that is “close” to x_0 to estimate $\hat{f}(x_0)$:

- ▶ this is appealing as we expect that the observations close to x_0 provide the most relevant information about $f(x_0)$,
- ▶ local models are often useful in situations where a suitable global has not been identified,
- ▶ types of smoothers differ in the way they use the data close to x_0 to estimate $\hat{f}(x_0)$.

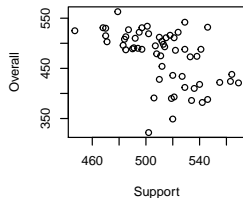
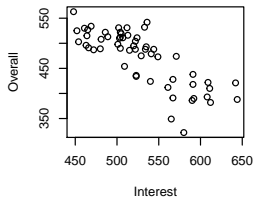
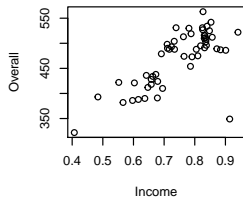
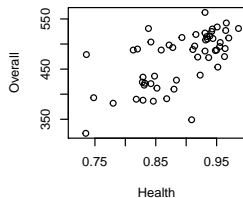
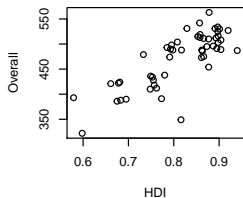
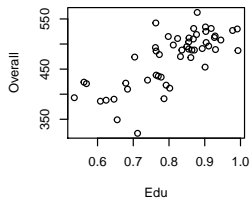
Example I



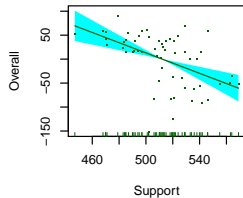
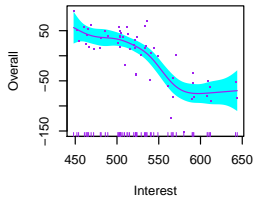
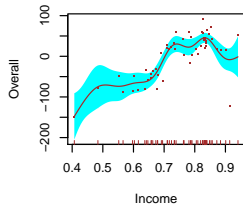
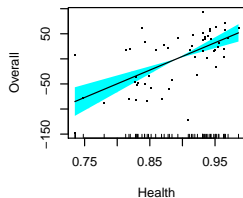
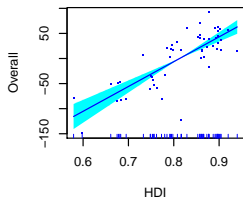
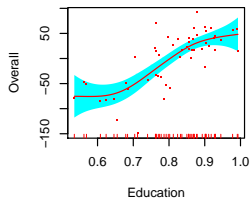
Example I



Recall PISA Example



Recall PISA Example



Three Classes of Smoothers

Some common smoothers include:

1. **Regression splines**.
2. **Kernel smoothers** (Nadaraya-Watson, locally weighted averages, local regression, loess). The functions `loess()` and `lm()` fit in this category. †
3. **Smoothing splines** (roughness penalties).

We will look at regression splines and smoothing splines.

Regression Splines I

Idea: Although a low order polynomial may not apply globally, it may be sensible to “stitch together” a series of local low order polynomial models.

Regression splines use a *piecewise polynomial* of usually low degree, e.g., 1 or 2 or 3. The regions are separated by *knots* (or *breakpoints*). The positions where each pair of *segments* join are called *joints*. The more knots, the more flexible the family of curves become.

It is customary to force the piecewise polynomials to join smoothly at these knots. A popular choice is piecewise cubic polynomials with continuous 0th, 1st and 2nd derivatives called *cubic splines*. Using splines of degree > 3 seldom yields any advantage.

Given a set of knots, the smooth is computed by multiple regression on a set of *basis functions*.

Regression Splines II

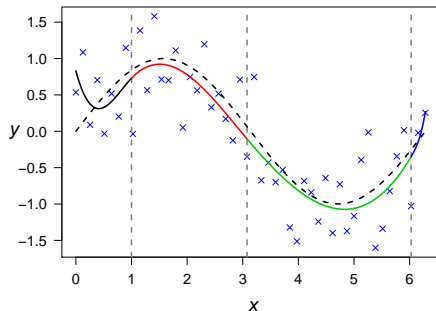


Figure: Smoothing some data with a regression spline (B-spline[†]). Each segment of the spline is coloured differently. The true function is the sine function (dashed) and $n = 50$.

Smoothing Splines - Example I

Rather than regressing on a set of basis functions, *cubic smoothing splines* are the solution to an optimization problem. They minimize over a space of smooth functions

$$\dagger S(f) = \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda \int_a^b \{f''(x)\}^2 dx, \quad (4)$$

where $a < x_1 < \dots < x_n < b$ for some a and b , and $\lambda \geq 0$.

The terms of $S(f)$:

1. The first penalizes lack-of-fit;
2. the second penalizes wiggleness.

These two conflicting quantities are weighted by the non-negative *smoothing parameter* λ .

Smoothing Splines - Example II

Here's an example from the **HRW** package.

The data set `SydneyRealEstate` contains the following variables (in addition to many more!) concerning houses sold in Sydney, Australia, during 2001:

- ▶ `logSalePrice`: the natural logarithm of sale price (\$AUD),
- ▶ `distToGPO`: the distance from the house to the General Post Office in Sydney's central business district (kilometers).

Smoothing Splines - Example III

```
> data(SydneyRealEstate, package = "HRW")
> dim(SydneyRealEstate)

[1] 37676    39

> set.seed(321) # Too large, so selecting a random sample
> index <- sample(nrow(SydneyRealEstate), size = 1000)
> smallsydney <- na.omit(SydneyRealEstate[index, ]) # na.omit() not needed
> dim(smallsydney)

[1] 1000    39

> with(smallsydney, length(unique(distToGP0))) # Any ties?

[1] 1000

> plot(logSalePrice ~ distToGP0, smallsydney,
       main = "Sydney property prices and smoothing splines")
> syd.spl <- with(smallsydney, smooth.spline(distToGP0, logSalePrice))
```

Smoothing Splines - Example IV

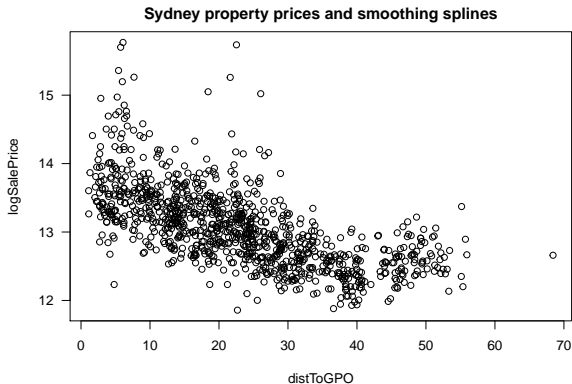


Figure: Scatterplot of a random sample of the `SydneyRealEstate` data in **HRW**.

Smoothing Splines - Example V

```
> syd.spl
```

Call:

```
smooth.spline(x = distToGP0, y = logSalePrice)
```

Smoothing Parameter spar= 0.678 lambda= 2.095e-05 (12 iterations)

Equivalent Degrees of Freedom (Df): 25.77

Penalized Criterion (RSS): 171.1

GCV: 0.5297

```
> plot(logSalePrice ~ distToGP0, smallsydney, col = "gray")
> # Automatic smoothing parameter selection via GCV:
> syd.spl <- with(smallsydney, smooth.spline(distToGP0, logSalePrice))
> lines(syd.spl, col = "blue", lwd = 2)
> with(smallsydney, lines(smooth.spline(distToGP0, logSalePrice, df = 5),
                           lty = 2, col = "red", lwd = 2))
> with(smallsydney, lines(distToGP0, col = "purple", lty = 2, lwd = 1,
                           fitted(lm(logSalePrice ~ distToGP0, smallsydney)))))
```

Smoothing Splines - Example VI

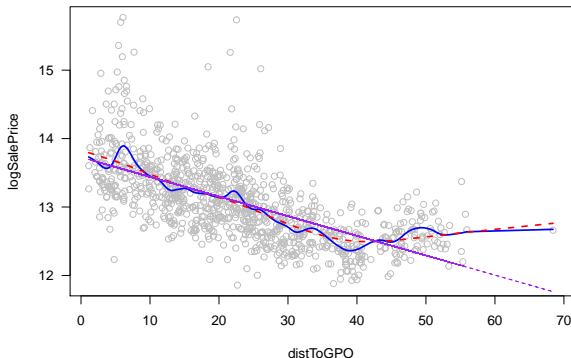


Figure: Example of `smooth.spline()` run on a random sample of the SydneyRealEstate data in HRW. **blue** curve: GCV ($DF=25.7$); **red** curve: smooth spline ($DF=5$); **purple** line: linear model. What statistical model might be appropriate?

Smoothing Splines - Example VII

Exercises

1. Write a few sentences on what you see in the scatter plot.
Does what you see make sense?
2. What price range in AUD is the response variable? Given the data were collected in 2001, how does it compare with Auckland currently?

EDF I

There are dozens of popular smoothers and they all allow the user to vary the amount of smoothing done via a smoothing parameter, e.g., **bandwidth**, the **span**, or λ . Undersmoothing and oversmoothing are common problems when smoothing: choosing a reasonable amount of smoothing *is the most important consideration*—much more so than the type of smoother.

It is useful to have some measure of the amount of smoothing done. One such measure is the *effective degrees of freedom (EDF)*. It is useful for a number of reasons, e.g., comparing different types of smoothers while keeping the amount of smoothing roughly equal. Technically, they only apply to *linear* smoothers.

EDF II

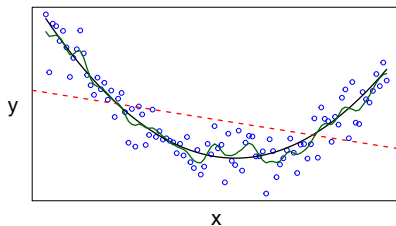
Very very crudely,

- ▶ an EDF of 1 is a linear function,
- ▶ an EDF of 2 has the fitting flexibility of a quadratic,
- ▶ an EDF of 3 has the fitting flexibility of a cubic. . .
- ▶ Actually, an EDF of 5 often has the wiggleness of a cubic or quartic.
- ▶ An EDF between 1 and 6 is often okay for many scatterplots.
- ▶ An EDF of 20 or more is usually far too much.

It is important to look carefully at your smooth and see if it underfits or overfits!

EDF III

The best model achieves a parsimonious balance between under- and over- fitting. Within a regression context, consider the figure:



Applying simple linear regression (**dashed line**) results in under-fitting. Applying a smoothing spline with many degrees of freedom (**wiggly solid line**) results in over-fitting (undersmoothing). The most parsimonious relationship here (solid line) is a quadratic.

CAUTION!

Some things to be aware of when using smoothers:

- ▶ It is important to get the level of smoothing correct.
- ▶ Often smoothers do not perform well at the edge of the data (or for extrapolation).
- ▶ Outliers and high leverage points may have a big impact on $\hat{f}(x_0)$ if they are in the data window for x_0 (otherwise, no problem!).
- ▶ They require larger amounts of data than global models. Also, they will not work well in regions where data is sparse.

GAMS vs SMOOTHERS

The key feature of a GAM is that it **automatically** selects appropriate smoothness.

Examples - Kauri Example I

Let Y = presence/absence of a tree species, *agaaus* (*Agathis australis*), also known as Kauri, from 392 sites from the Hunua forest. This data frame is found in **VGAM**. We will fit a nonparametric logistic regression.

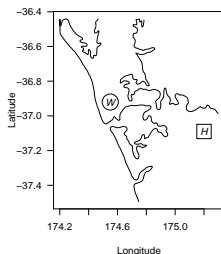


Figure: The Hunua and Waitakere Ranges.

Examples - Kauri Example II

```
> c(with(hunua, length(unique(altitude))), nrow(hunua))
```

```
[1] 64 392
```

```
library("mgcv")
gamfit.h1 <- gam(agaaus ~ s(altitude), family = binomial, data=hunua)
# summary(gamfit.h1)
plot(gamfit.h1, main="(a)")
# Sort the data by altitude---needed by plot (b) (effectively lines()):
ooo <- with(hunua, order(altitude))

plot(fitted(gamfit.h1)[ooo] ~ altitude[ooo], data = hunua,
     type = "l", ylim = c(0, 1), # ylim contains 1 for a reason!
     lwd = 2, col = "blue", main = "(b)",
     xlab = "altitude", ylab = "Fitted value")
with(hunua, points(altitude, jitter(agaaus, f = 0.1), col = "orange"))
```

Examples - Kauri Example III

The fitted model is

$\text{logit Pr}[Y_{\text{agaaus}} = 1] = f(\text{altitude}) = \beta_0 + f_1(x_1)$ where f is a smooth function determined from the data (here, we have absorbed the intercept into the smoother).

Tip: most GAM **R** packages use `s()` to denote a smooth, so remember to `detach()` package after use. Don't have two GAM **R** packages loaded at any one time else there will be problems! Use `search()` to see.

Examples - Kauri Example IV

```
> search()
```

```
[1] ".GlobalEnv"          "package:mgcv"          "package:nlme"  
[4] "package:readr"        "package:VGAMdata"      "package:VGAM"  
[7] "package:splines"      "package:stats4"        "package:knitr"  
[10] "package:stats"        "package:graphics"      "package:grDevices"  
[13] "package:utils"        "package:datasets"      "package:methods"  
[16] "Autoloads"           "package:base"
```

```
> detach("package:VGAMdata")
```

```
> detach("package:VGAM")
```

Examples - Kauri Example V

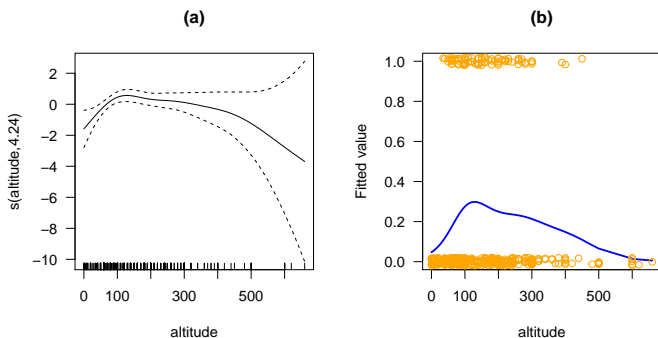


Figure: GAM plots of Kauri data in the Hunua ranges. (a) Fitted (centred) component function $\hat{f}_1(x_1)$, with pointwise ± 2 SE bands. (b) Fitted probabilities $\hat{P}(Y=1|\text{altitude})$ with the actual y_i plotted as jittered points.

Examples - Kauri Example VI

```
> summary(gamfit.h1)
```

Family: binomial

Link function: logit

Formula:

agaaus ~ s(altitude)

Parametric coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-1.42	0.14	-10.2	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:

	edf	Ref.df	Chi.sq	p-value
s(altitude)	4.24	5.24	12.3	0.035 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) = 0.0281 Deviance explained = 4.77%

UBRE = 0.0099896 Scale est. = 1 n = 392

Examples - Kauri Example VII

In contrast, a naive approach would be to fit:

```
> naivefit <- glm(agaas ~ altitude, binomial, data = hunua)
> coef(summary(naivefit))
```

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-1.2785488	0.205746	-6.2142	5.158e-10
altitude	-0.0002233	0.001027	-0.2173	8.280e-01

but this is a poor fit. A consequence of fitting a line on the logit scale is that one would conclude that the distribution of Kauri trees is not a function of altitude, which is biologically unrealistic! It has major lack-of-fit.

We will 'replace' `gamfit.h1` by a GLM that is more human-friendly—both in terms of understandability and usability. But we wouldn't have been able to unless we had first fitted the GAM.

Examples - Kauri Example VIII

We saw from the figure that there is strong evidence that Kauri response curve isn't linear in altitude on the logit scale. Now we will replace the model by a GLM, after an appropriate transformation of the covariate. We will call `altitude` x_1 for succinctness.

Some thoughts:

1. Looking at the GAM plot the (unimodal) curve might be a quadratic in altitude, however a quadratic is symmetric by nature and would be inadequate here because of the right skew.
2. So let's transform x_1 to symmetrize it so that a quadratic might be okay. The RHS is skewed, so we apply the ladder-of-powers to pull in the tail. So let's try x_1^α for some power $\alpha < 1$. (If we chose $\alpha = 0$ then that's equivalent to $\log x_1$.)

Examples - Kauri Example IX

3. The rugplot indicates that the *distribution* of altitude is right-skewed but that is not a major problem. The problem with a quadratic is that the smooth *function* is right-skewed, which is different.
4. Related to the previous point, the SE bands are wide, especially at the RHS because there is not much data there at high altitudes. Note that it is **false** to say that if you can fit a line through the bands then it is linear, because they are *pointwise* SE bands.
5. Let's try $\alpha = \frac{1}{2}$, i.e., a square root transformation. We will fit a quadratic in the transformed variable.

Examples - Kauri Example X

```
> shunua <- transform(hunua, sqrtalt = sqrt(altitude)) # New variable
> shunua <- shunua[order(with(shunua, altitude)), ] # Sort by altitude
> hfit31 <- gam(agaaus ~ s(sqrtalt), binomial, data = shunua)
> hfit41 <- glm(agaaus ~ poly(sqrtalt, 2, raw = TRUE), # sqrtalt + alt
               binomial, data = shunua, trace=FALSE)
> coef(hfit41)

               (Intercept) poly(sqrtalt, 2, raw = TRUE)1
                        -4.90537                        0.62713
poly(sqrtalt, 2, raw = TRUE)2
                        -0.02445
```

Now to show that `hfit41` is reasonable, let's plot the parametric and nonparametric fits together.

```
> plot(fitted(hfit31)~sqrtalt, shunua, type="l", col="darkred",
       xlab="Square root of altitude", ylab="Fitted values")
> lines(fitted(hfit31)~sqrtalt, shunua, lty=1, col="darkred")
> lines(fitted(hfit41)~sqrtalt, shunua, lty=2, col="blue")
> mycol=c("darkred", "blue")
> mylty=c(1, 2)
> legend("topright", col=mycol, lty=mylty, legend=c("GAM", "GLM"))
```

Examples - Kauri Example XI

This produces the figure below. The two fits are in reasonable agreement, especially at mid-altitudes where most of the data is.

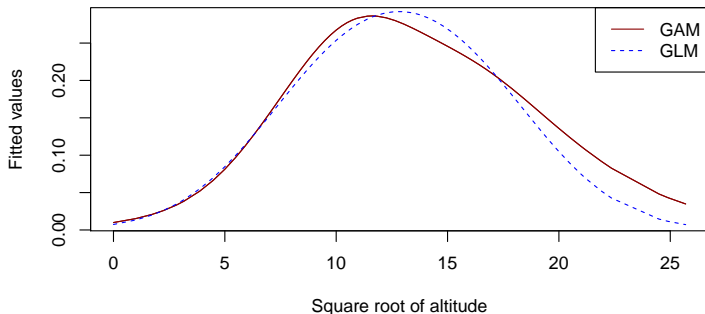


Figure: The fitted values of the two models.

Examples - Kauri Example XII

6. Thus fitting a quadratic in `sqrt(altitude)` is a good idea. The model is (see (6))

$$\text{logit Pr}(Y = 1) = \eta = \beta_0 + \beta_1 \sqrt{x_1} + \beta_2 x_1. \quad (5)$$

7. Some empirical evidence that this model is reasonable:

```
> round(coef(summary(hfit41)), 4)
```

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-4.9054	1.1285	-4.347	0.0000
poly(sqrtalt, 2, raw = TRUE)1	0.6271	0.1877	3.341	0.0008
poly(sqrtalt, 2, raw = TRUE)2	-0.0245	0.0075	-3.258	0.0011

shows that all coefficients are needed, i.e., are statistically significant.

Examples - Kauri Example XIII

8. Let's use AIC too to show that `hfit41` is better than `gamfit.h1`:

```
> c(AIC(hfit41), AIC(gamfit.h1))  
[1] 392.3 395.9
```

The transformed quadratic model has a smaller AIC.

9. Given our parametric model, let's extract some information from it that would be much more difficult to do if it were a GAM.

Firstly we can apply some basic maths: since

$$\begin{aligned}\text{logit } p(x_1) = \eta(x_1) &= \beta_0 + \beta_1 \sqrt{x_1} + \beta_2 (\sqrt{x_1})^2 \\ &= \beta_0 + \beta_1 \sqrt{x_1} + \beta_2 x_1\end{aligned}\quad (6)$$

Examples - Kauri Example XIV

where x_1 is altitude, then take the derivative with respect to x_1 :

$$\eta'(x_1) = \frac{1}{2} \frac{\beta_1}{\sqrt{x_1}} + \beta_2$$

which equals 0 when

$$\sqrt{x_1} = \frac{-\beta_1}{2\beta_2}, \text{ i.e., } x_1 = \frac{\beta_1^2}{4\beta_2^2}. \quad (7)$$

This is the location of the minimum/maximum of a quadratic.

[Useful: the slope (derivative) of ax^b wrt x is abx^{b-1} . †]

Thus

```
> opt.hunua <- as.vector((coef(hfit41)[2] / (2 * coef(hfit41)[3]))^2)
> opt.hunua # Est'd altitude where the max. prob. of presence occurs
[1] 164.4
```

Examples - Kauri Example XV

For checking purposes, the plot below plots the fitted probabilities against $\sqrt{x_1}$. Note the symmetry.

```
> plot(fitted(hfit41) ~ sqrtalt, shunua, type = "l", las = 1, main = "",  
       col = "blue", ylab = "Probability", xlab = expression(sqrt(x[1])))  
> abline(v = sqrt(opt.hunua), col = "orange", lty = "dashed")  
> abline(h = prob.max, col = "purple", lty = "dashed")
```

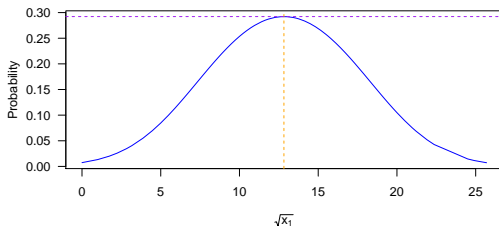


Figure: Probability of presence of Kauri in the Hunua ranges. The vertical line corresponds to the optimal altitude. The horizontal line corresponds to the optimal probability, `prob.max`.

Examples - Kauri Example XVI

10. Secondly, substituting (7) into (5), the maximum value of η is

$$\eta = \beta_0 - \frac{\beta_1^2}{2\beta_2} + \frac{\beta_1^2}{4\beta_2} = \beta_1 - \frac{\beta_2^2}{4\beta_3}. \quad (8)$$

Applying this to `hfit4` gives

```
> myvec <- coef(hfit41)
> (eta.max <- as.vector(myvec[1] - (myvec[2]^2) / (4 * myvec[3])))
[1] -0.8847
> (prob.max <- as.vector(logitlink(eta.max, inverse = TRUE)))
[1] 0.2922
```

This looks correct in the figure.

Examples - Mass Killings Example I

The magazine Mother Jones has recorded the number of mass-killings in the USA since 1982. Click [\[here\]](#) to visit this website:

This, from their website:

“Our research focused on indiscriminate rampages in public places resulting in four or more victims killed by the attacker. We exclude shootings stemming from more conventionally motivated crimes such as armed robbery or gang violence. Other news outlets and researchers have since published larger tallies that include a wide range of gun crimes in which four or more people have been either wounded or killed. While those larger data sets of multiple-victim shootings are useful for studying the broader problem of gun violence, our investigation provides an in-depth look at a distinct phenomenon—from the firearms used and mental health factors to the growing copycat problem. Tracking mass shootings is complex; we believe ours is the most useful approach.”

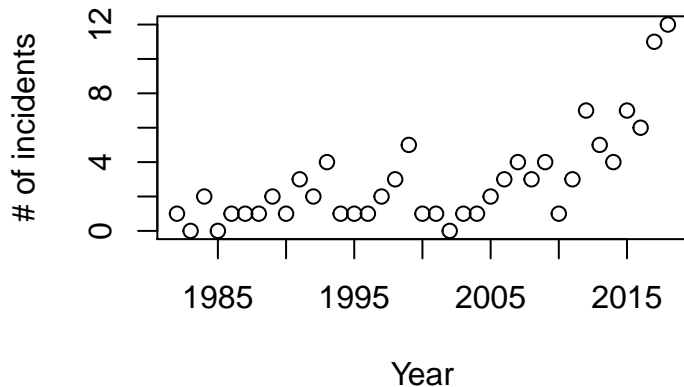
Examples - Mass Killings Example II

The data set `masskill.csv` contains the following variables:

- ▶ `year`: the year the incidents occurred,
- ▶ `popn`: the population of the USA for a given year (millions),
- ▶ `masskill`: the number incidents where a mass killing occurred.

First, plot the data:

Examples - Mass Killings Example III



Examples - Mass Killings Example IV

First we will fit a Poisson model, offset by population increase, where we assume that the log of the expected rate of mass killing incidents per 100 million people is thought to be linearly related to time (year):

```
masskill.lin <- glm(masskill ~ I(year-1982), family="poisson",  
                   offset=log(popn/1000), data=MK.df)
```

Now we will fit a Poisson model, again offset by population increase, where we assume that the log of the expected rate of mass killing incidents per 100 million people is thought to be quadratically related to time (year):

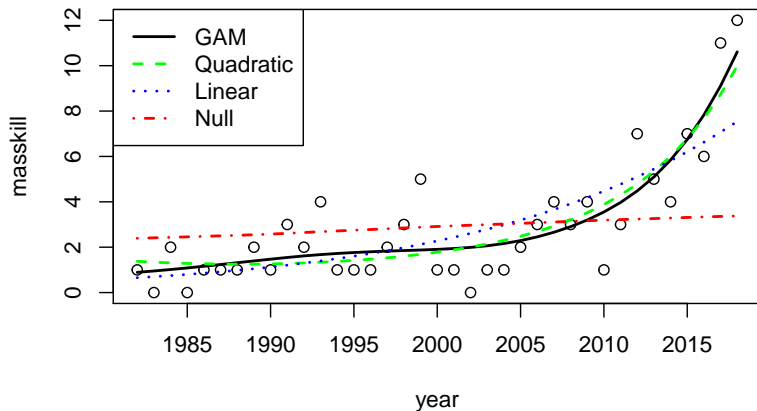
```
masskill.quad <- glm(masskill ~ I(year-1982)+(I(year-1982)^2),  
                    family="poisson", offset=log(popn/1000),  
                    data=MK.df)
```

Now fit a GAM:

```
masskill.gam <- gam(masskill ~ s(I(year-1982))+offset(log(popn/1000)),  
                   family="poisson", data=MK.df)
```

Examples - Mass Killings Example V

Mass killing by year – four models



Examples - Mass Killings Example VI

Checking goodness-of-fit statistics, and comparing the models:

Deviances: `1-pchisq(deviance,df)`

```
[1] 0.000394  
[1] 0.4014  
[1] 0.6084  
[1] 0.7516
```

AIC (also AICc, BIC):

	df	AIC
masskill.null	1.000	167.4
masskill.lin	2.000	134.4
masskill.quad	3.000	131.1
masskill.gam	3.817	128.9

Using GAM R Packages I

The following are popular specialized packages that fit GAMs:

- ▶ **gam** by Trevor Hastie at Stanford, is similar to the **S-PLUS** version.
- ▶ **mgcv** by Simon Wood at Bristol, has an emphasis on smoothing parameter selection. The most cutting edge implementation.
- ▶ **VGAM** from Auckland, fits 100+ models/distributions.

Tip: type `citation("gam")` etc. to cite the packages properly.

Here is some general information regarding the use of **mgcv**, **gam** and **VGAM**.

Using GAM R Packages II

- Use `s()` inside `gam()` or `vgam()` **only**, e.g.,

```
> fit1a = gam(y ~ s(x1) + s(x2) + x3, binomial, bdata) # mgcv
> fit1b = gam(y ~ s(x1) + s(x2, df = 1) + x3, binomial, bdata) # gam
> fit2 = vgam(y ~ s(x1) + s(x2, df = 1) + x3, binomialff, bdata) # VGAM
```

Object `fit1a` can be fitted in **mgcv**. Objects `fit1a` and `fit1b` can be fitted in **gam**. Object `fit2` runs only in **VGAM**. Setting `df = 1` makes the smooth function linear, hence the models may be written

- `fit1a`:

$$\text{logit Pr}(Y = 1) = \beta_0 + f_1(x_1) + f_2(x_2) + \beta_3 x_3$$

- `fit1b` and `fit2`:

$$\text{logit Pr}(Y = 1) = \beta_0 + f_1(x_1) + \beta_2 x_2 + \beta_3 x_3$$

One use of `fit1a` is to investigate the effect of `x3` after adjusting for `x1` and `x2`—but the user doesn't care what f_1 and f_2 look like.






Using GAM R Packages III

- ▶ It's not a good idea to use `s(z, df = 1)`; use `z` instead.
- ▶ Of course, one can only smooth continuous variables having enough unique values. Suggestion: don't try smoothing with less than 10 distinct values.
- ▶ Like all modelling function such as `lm()` and `glm()`, all variables should be inside a data frame. There should not be lots of vectors (one for each variable) floating around the workspace.
- ▶ Plotting the fitted component functions is a must. Using SE bands is often a good idea, and it is sometimes possible to constrain the *y*-axis of the plots to be on the same scale—it makes the functions more comparable.
- ▶ Try typing `help.start()` to read the online help.




Using GAM R Packages IV

- ▶ Many of the following generic functions available for `lm` objects are also available in GAM packages. They include:
 - ▶ `add1()` to add one term
 - ▶ `anova()` for statistical inference
 - ▶ `coef()`, $\hat{\beta}$, but not human interpretable for \hat{f}_k s
 - ▶ `deviance()` and `AIC()`
 - ▶ `df.residual()` is sometimes $n - p$ where p is the EDF,
 - ▶ `drop1()` to drop one term
 - ▶ `fitted()`, $\hat{y}_i = \hat{\mu}_i$
 - ▶ `model.matrix()`, \mathbf{X}
 - ▶ `plot()` for plotting component functions. Use!
 - ▶ `predict()`, $\hat{\eta}_i$, and at new data sets too
 - ▶ `print()`
 - ▶ `residuals()`, several types
 - ▶ `step()` for variable selection
 - ▶ `summary()`, $\hat{\beta}$, $\widehat{\text{Var}}(\hat{\beta})$, ...
 - ▶ `update()` as an easy method for adding or dropping terms.

References† I

-  Harezlak, J., Ruppert, D., Wand, M. P., 2018. Semiparametric Regression in R. Springer, New York, USA.
-  Hastie, T. J., Tibshirani, R. J., 1990. Generalized Additive Models. Chapman & Hall, London.
-  Perperoglou, A., Sauerbrei, W., Abrahamowicz, M., Schmid, M., 2019. A review of spline function procedures in R. BMC Medical Research Methodology 19 (1), 1–16.
-  Ruppert, D., Wand, M. P., Carroll, R. J., 2003. Semiparametric Regression. Cambridge University Press, Cambridge.
-  Wand, M. P., Ormerod, J. T., 2008. On semiparametric regression with O’Sullivan penalized splines. Australian & New Zealand Journal of Statistics 50 (2), 179–198.

References† II

-  Wood, S. N., 2006. Generalized Additive Models: An Introduction with R. Chapman and Hall, London.
-  Wood, S. N., 2017. Generalized Additive Models: An Introduction with R, 2nd Edition. Chapman & Hall/CRC, London.
-  Yee, T. W., 2015. Vector Generalized Linear and Additive Models: With an Implementation in R. Springer, New York, USA.