mgcv: GAMs in R

Simon Wood
Mathematical Sciences, University of Bath, U.K.

mgcv, gamm4

- mgcv is a package supplied with R for generalized additive modelling, including generalized additive mixed models.
- The main GAM fitting routine is gam.
- bam provides an alternative for very large datasets.
- The main GAMM fitting is gamm which uses PQL based on package nlme.
- gamm4 is an R package available from cran.r-project.org supplying gamm4, a version of gamm which uses lme4 for GAMM fitting, and avoids PQL. It is really an extension package for mgcv.
- The packages are loaded into R using e.g. library(mgcv).

`mgcv` help

- To get ‘overview’ help type `help.start()` within R and follow the `Packages` link to `mgcv`.
- The help page `mgcv-package` is a good page to start to get an overview.
- Note other overview pages such as `gam.models` and `gam.select`.
- All user visible functions are documented.
- Once a library is loaded its help pages are accessible via, e.g. `help("gam")` or `?gam`.
- Technical references on the underlying methods are given in `?gam` and via `citation("mgcv")`.
- Wood (2006) *Generalized additive models:an introduction with R* CRC/Taylor&Francis provides further information.

gam

- Use of the function `gam` is similar to the use of function `glm`, except for the following
  1. The model formula can contain smooth terms `s` and tensor product smooth terms `te` in the linear predictor.
  2. There are extra arguments controlling smoothing parameter estimation, notably `method` for choosing between `"REML"`, `"ML"`, `"GCV.Cp"` and `"GACV.Cp"` smoothness selection.
  3. The `family` argument can also be `Tweedie` or `negbin`.
- `gam` returns and object of class `"gam"`, which can be further interrogated using method functions such as `print`, `summary`, `anova`, `plot`, `predict`, `residuals` etc.
- The front end design of `gam` and its associated functions is based heavily on Trevor Hastie's original `gam` function for `S`. The underlying model representation and numerical methods are very different, however, being based on the penalized regression spline methods covered in this course.

`family` arguments usable with `gam`

- `gaussian` (default) is useful for real valued response data.
- `Gamma` is useful for strictly positive real valued data. The default link is only useful in some waiting time applications, and the log link is more often used.
- `poisson` is useful when the response is count data of some sort.
- `binomial` is used most often for binary (logistic) regression, but is applicable to any response that is the number of successes from a known number of trials.
- `inverse.gaussian` is for strictly positive real response variables: useful for various 'time to event' data.
- `quasi` does not define a full distribution, but allows inference when only the mean variance relationship can be well approximated. `quasipoisson` and `quasibinomial` are special cases. Not useable with likelihood based smoothness selection.
- `Tweedie` is an alternative to quasi when $\text{var}(y) = \phi\mu^p$, $1 < p < 2$, and a full distribution is required (for a non-negative real response).
- `negbin` is useful for overdispersed count data, but computation is slow.

Specifying models: some examples

- $y_i = f(x_i) + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma^2)$

  ```
  gam(y ~ s(x))
  ```

- $\log\{\mathbb{E}(y_i)\} = f_1(x_i) + f_2(z_i) + f_3(v_i) + w_i$ where $y_i \sim$ Poi.

  ```
  gam(y ~ s(x) + s(z) + s(v) + w,family=poisson)
  ```

- $\sqrt{\mathbb{E}(y_i)} = f_1(\text{time}_i, \text{distance}_i) + f_2(w_i)$, $y_i \sim$ gamma.

  ```
  gam(y ~ te(time,distance) + s(w),
      family=Gamma(link=sqrt))
  ```

  uses a scale invariant tensor product smooth for f_1.

- $\text{logit}\{\mathbb{E}(y_i)\} = w_i f_1(x_i, z_i) + f_2(v_i)$, $y_i \sim$ binary.

  ```
  gam(y ~ s(x,z,by=w) + s(v),family=binomial)
  ```

  Here f_1 is isotropic (a thin plate spline).

`s` term details

- `s(x,k=20,id=2,bs="tp")` is an example smooth specifier, used in a formula. (Some) arguments are ...

x	is the covariate of the smooth (can have any name!): some types of smooth can have several covariates (e.g. "tp").
bs	is the type of basis-penalty smoother.
k	is the basis dimension for the smooth (before imposing any identifiability constraints).
id	used to allow different smooths to be forced to use the same basis and smoothing parameter.
sp	allows the smoothing parameter to be supplied.
fx	if `TRUE` then the term is unpenalized.
by	allows specification of interactions of the smooth with a factor or metric variable.
m	specifies the penalty order for some bases.

Smooth classes

- Built in smooth classes (i.e. options for the `bs` argument of `s`) are:

`"cr"`	a penalized cubic regression spline ("cc" for cyclic version).
`"ps"`	Eilers and Marx style P-splines ("cp" for cyclic).
`"ad"`	adaptive smoothers based on "ps".
`"tp"`	Optimal low rank approximation to thin plate spline, any dimension and permissable penalty order is possible.

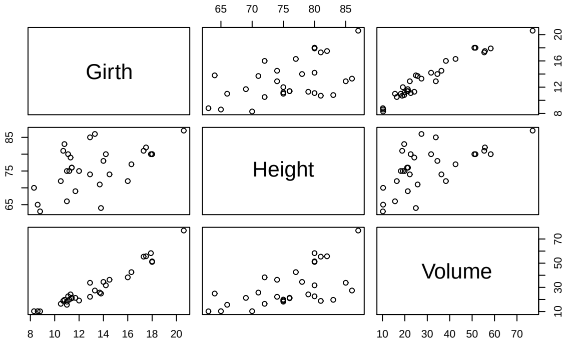
- In addition the `"re"` class implements simple random effects. For example `s(x,z,bs="re")` specifies a random effect $\mathbf{Zb}$ where $\mathbf{b} \sim N(\mathbf{0}, \mathbf{I}\sigma_b^2)$. $\mathbf{Z}$ is given by `model.matrix(~x:z-1)`. This approach is slow for large numbers od random effects, however.
- New classes can be added. See `?smooth.construct`

Tensor product smoothing in `mgcv`

- Tensor product smooths are constructed automatically from *marginal* smooths of lower dimension. The resulting smooth has a penalty for each marginal basis.
- `mgcv` can construct tensor product smooths from any *single penalty* smooths useable with `s` terms.
- `te` terms within the model formula invoke this construction. For example:
  - `te(x,z,v,bs="ps",k=5)` creates a tensor product smooth of x, z and v using rank 5 P-spline marginals: the resulting smooth has 3 penalties and basis dimension 125.
  - `te(x,z,t,bs=c("tp","cr"),d=c(2,1),k=(20,5))` creates a tensor product of an isotropic 2-D TPS with a 1-D smooth in time. The result is isotropic in x,z, has 2 penalties and a basis dimension of 100. This sort of smooth would be appropriate for a location-time interaction.

Simple data example

- To illustrate the basics of `gam`, consider a very simple dataset relating the timber volume of cherry trees to their height and trunk girth.



`trees` initial gam fit

- A possible model is

  $$\log(\mu_i) = f_1(\texttt{Height}_i) + f_2(\texttt{Girth}_i), \quad \texttt{Volume}_i \sim \text{Gamma}(\mu_i, \phi)$$

- Using rank 10 thin plate regression splines as the smoothers...

  ```
  library(mgcv)
  ct1 <- gam(Volume ~ s(Height) + s(Girth),
             family=Gamma(link=log),data=trees)
  ```

  estimates the model with default GCV smoothness selection.
- The results are stored in class `"gam"` object `ct1`.
- For the full contents of a `"gam"` object see `?gamObject`

print.gam

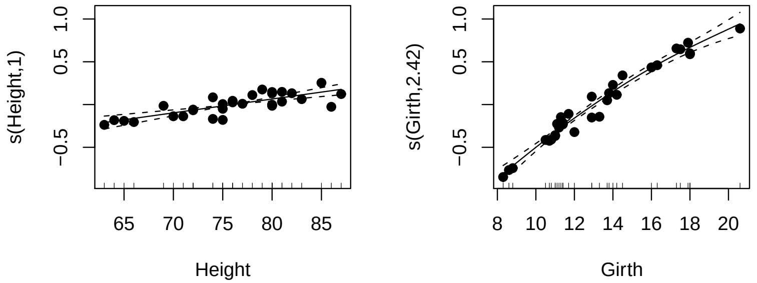
- Typing `ct1` causes R to pass `ct1` to the `print` method function.
- For class `"gam"` object `ct1` this means printing by `print.gam`.

```
> ct1

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth)

Estimated degrees of freedom:
1.0000 2.4222  total = 4.422254

GCV score: 0.008082356
```

- Notice how the EDFs for each term and the GCV score are reported.

plot.gam

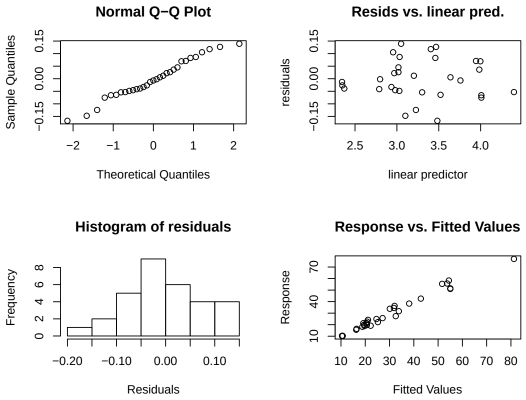
-
  ```
  par(mfrow=c(1,2))
  plot(ct1,residuals=TRUE,pch=19) ## calls plot.gam
  ```



- *partial* residuals for f_j are *weighted working residuals* from PIRLS added to $\hat{f}_j$. Systematic departure from f_j indicates a problem.
- Rug plot shows values of predictors.
- EDF for term reported in y axis lable.
- 95% Bayesian CIs shown (constraint causes vanishing CI on left).

Basic model checking: `gam.check`

-
  ```
  > gam.check(ct1) ## note QQ beefed up for next mgcv version
  ## smoothness selection convergence info omitted
  ```

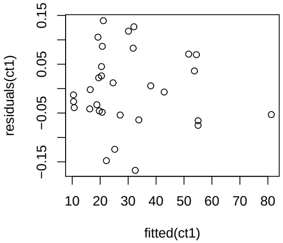


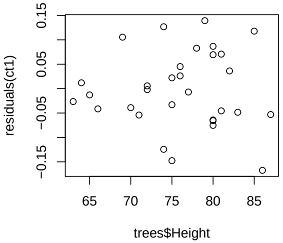
- *Deviance* residuals are used: often approximately normal.
- Plots are utterly useless for binary data!

residuals

- Other residual plots should be examined. e.g.

```
plot(fitted(ct1),residuals(ct1))
plot(trees$Height,residuals(ct1))
```





- ... OK, but possibly the short trees are less variable than Gamma is suggesting.
- Alternatives for `residual` argument `type` are `"deviance"` (default), `"pearson"`,`"scaled.pearson"`,`"working"` and (the unstandardized) `"response"`.

Checking basis dimensions k

- Can simply increase suspect k and see if fit measures change. . .

```
> gam(Volume~s(Height)+s(Girth,k=20),
+            family=Gamma(link=log),data=trees)

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth, k = 20) ## increase k from 10 to 20

Estimated degrees of freedom:
1.0000 2.4248  total = 4.424817

GCV score: 0.008083182
```

- . . . no evidence for change in fit.
- Note that a 20 dimensional basis includes a larger space of EDF 2.5 functions, than a 10 dimensional basis. . .

Cheaper `k` check

- Refitting complex models with increased `k` can be costly.
- A cheaper alternative is to check the adequacy of `k` by checking for unmodelled pattern in the residuals. . .

```
> rsd <- residuals(ct1,type="deviance")
> gam(rsd~s(Girth,k=20)-1,data=trees,select=TRUE)

Family: gaussian
Link function: identity

Formula:
rsd ~ s(Girth, k = 20) - 1

Estimated degrees of freedom:
4.839e-09  total = 4.839038e-09

GCV score: 0.005940881
```

- . . . no signal detected $\Rightarrow$ `k` large enough for `Girth`! (`select` allows 0 model to be selected, `-1` suppresses intercept).

Is smoothness selection robust?

- To check robustness of smoothness selection, fit with an alternative smoothness selection criterion. e.g.

```
> ct1 <- gam(Volume~s(Height)+s(Girth),
+             family=Gamma(link=log),data=trees,method="ML")
> ct1
...
Formula:
Volume ~ s(Height) + s(Girth)

Estimated degrees of freedom:
1.0000 2.5097  total = 4.509708

ML score: 69.64346
```

- . . . EDFs much the same.
- A very different criterion is best (i.e. `"ML"` or `"REML"` versus `"GCV.Cp"` or `"GACV.Cp"`).
- Remember:likelihood based methods tend to be more robust.

summary.gam

- Once checking suggests that the model is acceptable, then we can proceed to more formal inference. e.g.

-
  ```
  > summary(ct1)
  ...
  Parametric coefficients:
              Estimate Std. Error t value Pr(>|t|)
  (Intercept)  3.27568    0.01493   219.4   <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(Height) 1.00  1.000  31.04 7.04e-06 ***
  s(Girth)  2.51  3.150 211.93  < 2e-16 ***
  ---
  Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

  R-sq.(adj) =  0.974   Deviance explained = 97.8%
  ML score = 69.643  Scale est. = 0.0069092  n = 31
  ```

anova.gam

- If you must have p-values, then anova is better for any model containing factor variables

```
> anova(ct1)
Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth)

Approximate significance of smooth terms:
            edf Ref.df      F  p-value
s(Height) 1.00   1.00  31.04 7.04e-06
s(Girth)  2.51   3.15 211.93  < 2e-16
```

$\hat{\beta}$: coef and vcov

- The estimated coefficients, $\hat{\beta}$, and the Bayesian covariance matrix for $\beta|\mathbf{y}$...

```
> b <- coef(ct1) ## estimated coefficients
> V <- vcov(ct1) ## Bayesian cov matrix
> b[1:4]         ## first 4 coefs
  (Intercept)   s(Height).1   s(Height).2   s(Height).3
 3.275684e+00 -1.449735e-07  1.936084e-07 -8.372687e-08
> V[1:4,1:4]
              (Intercept)   s(Height).1   s(Height).2   s(Height).3
(Intercept)  2.228777e-04  1.938385e-23 -5.369032e-23  7.371339e-24
s(Height).1  1.938385e-23  1.172984e-08  1.944762e-09 -1.629145e-09
s(Height).2 -5.369032e-23  1.944762e-09  2.797615e-08 -6.769112e-09
s(Height).3  7.371339e-24 -1.629145e-09 -6.769112e-09  3.914123e-09
```

- `vcov(ct1,freq=TRUE)` extracts the frequentist covariance matrix for $\hat{\beta}$.

The smoothing parameters $\boldsymbol{\lambda}$

- We can extract the λ_i estimates, and, for RE/ML smoothness selection, the covariance matrix of the log λ_i estimates.

```
> ct1$sp
   s(Height)     s(Girth)
5.838785e+04 2.164997e-01
> sp.vcov(ct1)
              [,1]       [,2]        [,3]
[1,]  3.596675e+04 -0.1227099 0.008221598
[2,] -1.227099e-01  1.4489860 0.103279838
[3,]  8.221598e-03  0.1032798 0.071741386
```

- Alternatively, extract information as (root) variance components . . .

```
> gam.vcomp(ct1)

Standard deviations and 0.95 confidence intervals:

              std.dev        lower        upper
s(Height) 1.175867e-05 2.267390e-86 6.098041e+75
s(Girth)  1.748033e-02 5.683986e-03 5.375839e-02
scale     7.969330e-02 6.129519e-02 1.036137e-01

Rank: 3/3
```

Predicting at new covariate values: `predict.gam`

- `predict(ct1)` returns the linear predictor corresponding to the original data.
- `predict.gam`'s main use is to predict from the model, given new values for the predictor variables...

```
> ## create dataframe of new values...
> pd <- data.frame(Height=c(75,80),Girth=c(12,13))
> predict(ct1,newdata=pd)
       1        2
3.101496 3.340104 ## model predictions (linear predictor scale)
```

- `predict` has several useful arguments. e.g.

```
> predict(ct1,newdata=pd,se=TRUE)
$fit
       1        2
3.101496 3.340104
$se.fit
         1          2
0.02057014 0.02453761
```

More predict.gam

- Predictions can also be returned by model term...

```
> predict(ct1,newdata=pd,se=TRUE,type="terms")
$fit
    s(Height)      s(Girth)
1 -0.01616144 -0.1580258499
2  0.06464663 -0.0002267264

$se.fit
    s(Height)   s(Girth)
1 0.002901765 0.01407485
2 0.011603897 0.01617026

attr(,"constant")
(Intercept)
   3.275684
```

- ...or on the response scale: `predict(ct1,pd,type="response")`.

Linear prediction matrices: `lpmatrix`

- `predict.gam` can return the matrix mapping the estimated coefficients to the linear predictor. e.g.

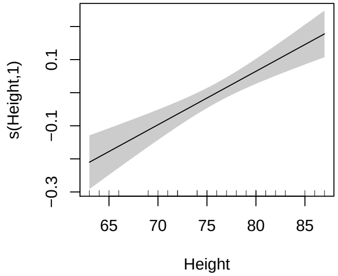
```
> Xp <- predict(ct1,newdata=pd,type="lpmatrix")
> Xp%*%coef(ct1) ## result same as predict(ct1,pd)
      [,1]
1 3.101496
2 3.340104
> Xp%*%vcov(ct1)%*%t(Xp) ## Bayesian cov matrix for predictions.
             1            2
1 0.0004231306 0.0003617439
2 0.0003617439 0.0006020943
```

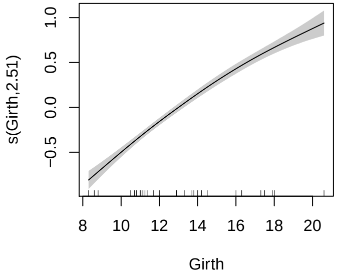
- Obviously `predict(ct1,type="lpmatrix")` returns the original model matrix used for fitting (take a look at the code for `model.matrix.gam` sometime).
- Linear predictor matrices are the key to inference about *any* quantity that can be predicted by the model. More later.

More visualization: customizing `plot.gam`

- `plot.gam` has quite a few options. Here are three examples
  1. The component-wise CIs have good coverage, except when smooths are close to straight lines. A solution is to include the uncertainty in the intercept when plotting.
  2. Each smooth can have its own y axis scale.
  3. Some people like their confidence regions to be shaded...

```
par(mfrow=c(1,2))
plot(ct1,shade=TRUE,seWithMean=TRUE,scale=0)
```

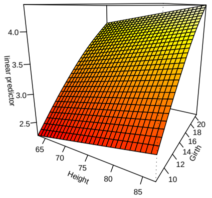


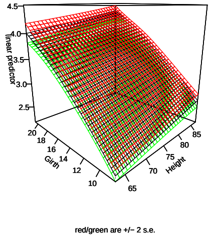


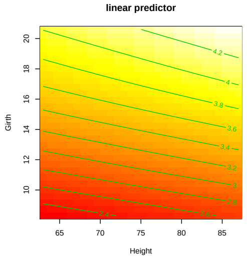
More visualization: `vis.gam`

- Sometimes it is helpful to see how the linear predictor or expected response would vary with 2 predictors, if all the others were held fixed at some value. `vis.gam` allows this.

-
  ```
  vis.gam(ct1,theta=30,ticktype="detailed")
  vis.gam(ct1,theta=-45,ticktype="detailed",se=2)
  vis.gam(ct1,plot.type="contour")
  ```







Alternative tree models

- Would a 2D smooth of Height and Girth be better?

```
> ct2 <- gam(Volume~s(Height,Girth,k=25),
+            family=Gamma(link=log),data=trees,method="ML")
> ct2
...
Estimated degrees of freedom:
4.3815  total = 5.381507

ML score: 71.01151
> AIC(ct1,ct2)
          df      AIC
ct1 5.509708 142.8381
ct2 6.381507 147.0865
```

- ...apparently not. Model has more degrees of freedom for a lower marginal likelihood, and the AIC is worse than before.
- But an isotropic smooth is not really appropriate here.

A tensor product smooth for trees

-
  ```
  > ct2 <- gam(Volume~te(Height,Girth),
  +             family=Gamma(link=log),data=trees,method="ML")
  > ct1;ct2
  ...
  Estimated degrees of freedom:
  1.0000 2.5097  total = 4.509708
  ML score: 69.64346
  ...
  Estimated degrees of freedom:
  3  total = 4.000024
  ML score: 66.71353
  > AIC(ct1,ct2)
            df      AIC
  ct1 5.509708 142.8381
  ct2 5.000024 143.4271
  > 2*66.71353+10 ## "AIC" ct2
  [1] 143.4271
  > 2*69.64346+8  ## "AIC" ct1
  [1] 147.2869
  ```
- . . . this is weird . . .

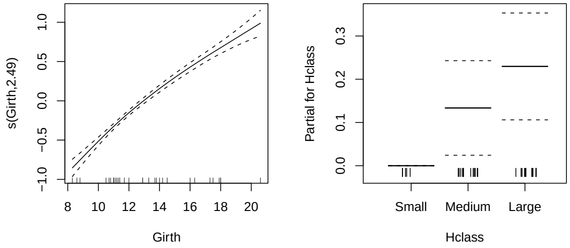
Which AIC is correct?

- `AIC(ct1,ct2)` selects the additive model, `ct1`. But `ct1` has lower marginal likelihood *and* more degrees of freedom, implying a *higher* AIC. Consider the two AIC computations. . .
  1. `AIC(ct1,ct2)` is based on the log likelihood of $\hat{\beta}$ treated as fixed effects, but with the model EDF in place of $\dim(\beta)$.
  2. Alternatively, base AIC on the marginal likelihood of the fixed parameters of the model with the random effect components of the smooths integrated out. Then the appropriate degrees of freedom is the number of fixed effects.
- Why do the 2 give different answers and which is 'right'?
- The difference lies in the model assumed. For option 1, we are being Bayesian, and would expect broadly the same smooths on resampling of the data. For option 2 we are implicitly being fully frequentist and assuming that the smooths would look completely different on resampling.
- So, use option 2 if *really* believe its model. I don't.

Other alternative models

- Having rejected a 2D smooth model, let's consider a couple of simple alternatives, based on discretizing `Height`.

```
trees$Hclass <- factor(floor(trees$Height/10)-5,
                      labels=c("Small","Medium","Large"))
ct3 <- gam(Volume~s(Girth)+Hclass,
           family=Gamma(link=log),data=trees,method="ML")
par(mfrow=c(1,2))
plot(ct3,all.terms=TRUE)
```



- Again AIC etc, suggest that this is no improvement.

A random effect for `Height`

- Finally try $\log \mu_i = f(\texttt{Girth}_i) + b_k$ if tree i is in height class k. The b_k are i.i.d. $N(0, \sigma_b^2)$ and $\texttt{Volume}_i \sim \text{Gamma}(\mu_i, \phi)$.

```
> ct4 <- gam(Volume~s(Girth)+s(Hclass,bs="re"),
+            family=Gamma(link=log),data=trees,method="ML")
> ct4
...
Estimated degrees of freedom:
2.4876 1.6294  total = 5.117026

ML score: 79.12945
> gam.vcomp(ct4)

Standard deviations and 0.95 confidence intervals:

              std.dev      lower     upper
s(Girth)  0.02108980 0.00615994 0.0722052
s(Hclass) 0.07889864 0.02726095 0.2283484
scale     0.09911287 0.07492786 0.1311042
```

- Again AIC etc suggest that this is a worse model.

Summary

- gam is like glm, but with extra facilities to allow smooth functions of covariates in the linear predictor.
- Residual checking is as for a GLM, but plot.gam plots the smooths, not residual plots.
- Additional model checking should check the smoothing basis dimension choice, and the reasonableness of the smoothness selection.
- Sometimes model selection is similar to model selection for ordinary GLMs.