

Model selection with **MuMIn** and **GAMM**

Kamil Bartoń

August 6, 2011

1 Extending **MuMIn**'s functionality to support **gamm**

The two principal functions in **MuMIn**, `model.avg` and `dredge` rely on availability of methods for a several generic function for the class of the given fitted model object. These generic functions include ones defined in package **stats** (`logLik`, `formula`, `nobs`, and optionally `deviance` which may simply return `NULL`), as well as ones defined in **MuMIn** itself (`coeffs`, `getAllTerms` and `tTable`). In some cases the default methods may work as well.

In case of **gamm** and **gamm4**, the returned object has no special class, it is a list with two items: `lme` or `mer`, and `gam` (with some information stripped from it). Therefore no specific methods can be applied.

The solution is to provide a wrapper function for **gamm** that evaluates the model and adds a class attribute onto it, e.g.:

```
> gamm <- function(...) structure(c(mgcv::gamm(...), list(call = match.call())),  
+   class = c("gamm", "list"))
```

similarly for **gamm4** (but assign the same class **gamm**):

```
> gamm4 <- function(...) structure(c(gamm4::gamm4(...), list(call = match.call())),  
+   class = c("gamm", "list"))
```

As they have the same names as the actual functions, so it is invisible for the user, and masks the original functions on the level of `.GlobalEnv`.

In addition, the wrappers add a `call` element, containing the original call to the wrapper function. It is not necessary, but makes things easier later on for `dredge`.

Once we have an object of class **gamm**, it is possible to provide methods for it. First let us define the generic methods from **stats**.

```
> logLik.gamm <- function(object, ...) logLik(object[[if (is.null(object$lme)) "mer" else "lme"]],  
+   ...)  
> formula.gamm <- function(x, ...) formula(x$gam, ...)  
> nobs.gamm <- function(object, ...) nobs(object$gam, ...)
```

It should be noted here that the issue of what the log-likelihood for **GAMM** should be is not entirely clear. The documentation for **gamm** states that the log-likelihood of `lme` is not the one of the fitted **GAMM**. However, comparing alternative models shows some evidence that it may be still appropriate for **gamm**. Namely the log-likelihood of fitted `lme`, and one of the `lme` part of **gamm** (including only linear terms to make the comparison adequate), have identical values.

```

> dat <- gamSim(6, n = 100, scale = 0.2, dist = "gaussian")

4 term additive + random effectGu & Wahba 4 term additive model

> fm1 <- gamm(y ~ x0 + x1 + x2 + x3, data = dat, random = list(fac = ~1),
+   method = "ML")
> fm2 <- lme(y ~ x0 + x1 + x2 + x3, data = dat, random = list(fac = ~1),
+   method = "ML")
> logLik(fm1$lme)

'log Lik.' -214.5197 (df=7)

> logLik(fm2)

'log Lik.' -214.5197 (df=7)

```

Likewise is in the generalised case of `gamm4` and `lmer`:

```

> dat <- gamSim(6, n = 100, scale = 0.2, dist = "poisson")

4 term additive + random effectGu & Wahba 4 term additive model

> fmg1 <- gamm4(y ~ x0 + x1 + x2 + x3, family = poisson, data = dat,
+   random = ~(1 | fac))
> fmg2 <- lmer(y ~ x0 + x1 + x2 + x3 + (1 | fac), family = poisson,
+   data = dat)
> logLik(fmg1$lmer)

'log Lik.' -460.5087 (df=6)

> logLik(fmg2)

'log Lik.' -460.5087 (df=6)

```

Similarly, comparison of `gamm4` with a smooth term, with fixed two degrees of freedom gives log-likelihood which is very close to that of `lmer` that includes a linear and quadratic term.

```

> fmgs1 <- gamm4(y ~ x0 + s(x1, k = 3, fx = TRUE) + x2 + x3, family = poisson,
+   data = dat, random = ~(1 | fac))
> fmgs2 <- lmer(y ~ x0 + x1 + I(x1^2) + x2 + x3 + (1 | fac), family = poisson,
+   data = dat)
> logLik(fmgs1$lmer)

'log Lik.' -459.4854 (df=7)

> logLik(fmgs2)

'log Lik.' -460.3622 (df=7)

```

Normally, the object returned by `gam` inherits also from `glm`, so the `nobs` method for `glm` is called, but in case of `gamm` the `gam` element has only class `gam`, so we need to define method directly (it just calls `nobs.glm`):

```
> nobs.gam <- function(object, ...) stats::nobs.glm(object, ...)
```

Methods for generic functions defined in MuMIn:

```
> coeffs.gamm <- function(model) coef(model$gam)
> getAllTerms.gamm <- function(x, ...) getAllTerms(x$gam)
> tTable.gamm <- function(model, ...) tTable(model$gam)
```

(the name `tTable` is somewhat misleading, as the `data.frame` returned does not need to contain *t*-values, two columns are obligatory: ‘Estimate’ and ‘Std. Error’)

2 Model selection

Now we have all the prerequisites to proceed with the model selection:

```
> fmgs1 <- gamm4(y ~ s(x0) + s(x1) + s(x2) + s(x3), family = poisson,
+   data = dat, random = ~(1 | fac))
> (dd <- dredge(fmgs1))
```

```
Global model: gamm4(y ~ s(x0) + s(x1) + s(x2) + s(x3), family = poisson, data = dat,
  random = ~(1 | fac))
```

Model selection table

	(Int)	s(x0)	s(x1)	s(x2)	s(x3)	k	AICc	delta	weight
8	3.097	+	+	+		8	197.6	0.000	0.901
16	3.096	+	+	+	+	10	202.0	4.427	0.099
7	3.119		+	+		6	258.5	60.930	0.000
15	3.117		+	+	+	8	264.5	66.910	0.000
14	3.123	+		+	+	8	526.7	329.200	0.000
13	3.136			+	+	6	541.4	343.800	0.000
6	3.139	+		+		6	546.5	348.900	0.000
5	3.154			+		4	570.4	372.800	0.000
12	3.148	+	+		+	8	698.4	500.800	0.000
4	3.189	+	+			6	911.5	713.900	0.000
11	3.200		+		+	6	1007.0	809.800	0.000
10	3.209	+			+	6	1153.0	955.000	0.000
3	3.237		+			4	1210.0	1012.000	0.000
2	3.249	+				4	1315.0	1117.000	0.000
9	3.261				+	4	1410.0	1213.000	0.000
1	3.304					2	1624.0	1426.000	0.000

```
> summary(model.avg(dd, subset = delta <= 8))
```

Call: model.avg(object = dd, subset = delta <= 8)

Model summary:

	Deviance	AICc	Delta	Weight
1+2+3	197.56	0.00	0.9	
1+2+3+4	201.99	4.43	0.1	

Variables:

1	2	3	4
s(x0)	s(x1)	s(x2)	s(x3)

Model-averaged coefficients:

	Coefficient	SE	z	value	Pr(> z)	
(Intercept)	3.097e+00	3.174e-01	9.758	< 2e-16	***	
s(x0).1	-2.940e-01	1.109e-01	2.651	0.008028	**	
s(x0).2	-1.515e-01	3.174e-01	0.477	0.633289		
s(x0).3	9.648e-03	7.350e-02	0.131	0.895559		
s(x0).4	-1.059e-01	1.754e-01	0.604	0.545859		
s(x0).5	3.116e-02	5.853e-02	0.532	0.594481		
s(x0).6	-1.395e-01	1.584e-01	0.880	0.378622		
s(x0).7	5.167e-02	6.826e-02	0.757	0.449082		
s(x0).8	5.924e-01	3.963e-01	1.495	0.134994		
s(x0).9	2.113e-01	1.749e-01	1.208	0.227065		
s(x1).1	9.901e-03	3.950e-02	0.251	0.802062		
s(x1).2	1.714e-02	6.347e-02	0.270	0.787079		
s(x1).3	5.271e-03	2.023e-02	0.261	0.794462		
s(x1).4	1.647e-02	3.449e-02	0.477	0.633094		
s(x1).5	-7.000e-03	1.422e-02	0.492	0.622478		
s(x1).6	-1.773e-02	3.088e-02	0.574	0.565908		
s(x1).7	-2.885e-03	8.010e-03	0.360	0.718730		
s(x1).8	-9.510e-02	1.032e-01	0.922	0.356607		
s(x1).9	3.805e-01	5.020e-02	7.581	< 2e-16	***	
s(x2).1	1.089e+00	1.952e-01	5.578	< 2e-16	***	
s(x2).2	-2.615e+00	6.846e-01	3.819	0.000134	***	
s(x2).3	-1.712e+00	2.503e-01	6.843	< 2e-16	***	
s(x2).4	-4.566e-01	4.323e-01	1.056	0.290877		
s(x2).5	1.995e-01	1.751e-01	1.139	0.254537		
s(x2).6	-9.772e-01	4.896e-01	1.996	0.045955	*	
s(x2).7	-4.775e-02	2.428e-01	0.197	0.844070		
s(x2).8	3.361e+00	1.074e+00	3.128	0.001758	**	
s(x2).9	1.162e+00	4.706e-01	2.470	0.013527	*	
s(x3).1	0.000e+00	4.560e-07	0.000	1.000000		
s(x3).2	6.407e-36	7.213e-07	0.000	1.000000		
s(x3).3	5.520e-37	1.428e-07	0.000	1.000000		
s(x3).4	-3.223e-36	4.225e-07	0.000	1.000000		

s(x3).5	5.615e-37	1.276e-07	0.000	1.000000
s(x3).6	-2.195e-36	3.703e-07	0.000	1.000000
s(x3).7	1.093e-36	1.875e-07	0.000	1.000000
s(x3).8	-1.437e-20	1.266e-06	0.000	1.000000
s(x3).9	1.391e-03	7.738e-03	0.180	0.857333

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

Non-present predictors taken to be zero

Relative variable importance:

s(x0)	s(x1)	s(x2)	s(x3)
1.0	1.0	1.0	0.1