6.20 The `xij` Argument

The following is valid for VGAM 0.7-9 and higher. Prior versions of VGAM handled the `xij` argument differently (in a messy and an inferior way)—so be warned!

6.20.1 Introduction

In all of the above VGAM documentation, we have had

$$\eta_j(\boldsymbol{x}_i) \;=\; \boldsymbol{\beta}_j^T \boldsymbol{x}_i \;=\; \sum_{k=1}^{p} x_{ik}\, \beta_{(j)k} \tag{6.39}$$

as the jth linear predictor $(j = 1, \ldots, M)$. Importantly, this can be generalized to

$$\eta_j(\boldsymbol{x}_{ij}) \;=\; \boldsymbol{\beta}_j^T \boldsymbol{x}_{ij} \;=\; \sum_{k=1}^{p} x_{ikj}\, \beta_{(j)k}, \tag{6.40}$$

or writing this another way (as a mixture),

$$\eta_j(\boldsymbol{x}_i^*, \boldsymbol{x}_{ij}^*) \;=\; \boldsymbol{\beta}_j^{*T} \boldsymbol{x}_i^* + \boldsymbol{\beta}_j^{**T} \boldsymbol{x}_{ij}^*. \tag{6.41}$$

Often $\boldsymbol{\beta}_j^{**} = \boldsymbol{\beta}^{**}$, say. In (6.41) the variables in $\boldsymbol{x}_i^*$ are common to all η_j, and the variables in $\boldsymbol{x}_{ij}^*$ have different values for differing η_j. This allows for covariate values that are specific to each η_j, a facility which is very important in many applications. Here are two simple examples.

1. Suppose there are two binary responses, $Y_j = 1$ or 0 for presence/absence of a disease in the jth eye, where $j = 1, 2$ for the left and right eye respectively. There is a single covariate, called ocular pressure, which measures the internal fluid pressure within each eye. With data from n people, it would be natural to fit an exchangeable bivariate logistic model:

$$\begin{aligned} \text{logit}\, P(Y_{ij} = 1) &= \beta_{(1)1}^* + \beta_{(1)2}^*\, x_{i2j}, \quad j = 1, 2; \quad i = 1, \ldots, n; \\ \log\, \psi &= \beta_{(2)1}^*, \end{aligned} \tag{6.42}$$

   where the dependency between the responses is modelled through the odds ratio ψ (here, it is modelled using only an intercept term). Note that the regression coefficient for x_{i21} and x_{i22} is the same, and $x_{i21} \neq x_{i22}$ in general because each person's eye will usually have a different ocular pressure. The constraint matrices are $\mathbf{H}_1 = \mathbf{I}_3$ and $\mathbf{H}_2 = (1, 1, 0)^T$, and they can be set up with `family = binom2.or(exchangeable=TRUE, zero=3)`.

2. Suppose an econometrician is interested in peoples' choice of transport for travelling to work and that there are four choices: $Y = 1$ for "bus", $Y = 2$ "train", $Y = 3$ "car" and $Y = 4$ means "walking". Assume that people only choose one means to go to work. Suppose there are three covariates: $X_2 =$ cost, $X_3 =$ journey time, and $X_4 =$ distance. Of the covariates only X_4 (and the intercept X_1) is the same for all transport choices; the cost and journey time differ according to the means chosen. Suppose a random sample of n people is collected from some population, and that each person has access to all these transport modes[4]. For such data, a natural regression model would be a multinomial logit model with $M = 3$: for $j = 1, \ldots, M$, $\eta_j =$

$$\log \frac{P(Y = j)}{P(Y = M + 1)} \;=\; \beta_{(j)1}^* + \beta_{(1)2}^*\, (x_{i2j} - x_{i24}) + \beta_{(1)3}^*\, (x_{i3j} - x_{i34}) + \beta_{(1)4}^*\, x_{i4}, \tag{6.43}$$

[4] If not then this is known as a "varying choice set" in the discrete choice model literature. Unfortunately this resides outside the VGLM/VGAM framework.

where, for the ith person, x_{i2j} is the cost for the jth transport means, and x_{i3j} is the journey time of the jth transport means. The distance to get to work is x_{i4}; it has the same value regardless of the transport means.

Equation (6.43) implies $\mathbf{H}_1 = \mathbf{I}_3$ and $\mathbf{H}_2 = \mathbf{H}_3 = \mathbf{H}_4 = \mathbf{1}_3$. Note also that if the last response category is used as the baseline or reference group (the default of `multinomial()`) then $x_{ik,M+1}$ can be subtracted from x_{ikj} for $j = 1, \ldots, M$—this is the natural way $x_{ik,M+1}$ enters into the model.

The use of the `xij` argument with the VGAM family function `multinomial()` has a very important application in economics with *consumer choice* or *discrete choice* modelling. In that field the term "multinomial logit model" includes a variety of models such as the "generalized logit model" where (6.39) holds, the "conditional logit model" where (6.40) holds, and the "mixed logit model," which is a combination of the two, where (6.41) holds. The generalized logit model focusses on the individual as the unit of analysis, and uses individual characteristics as explanatory variables, e.g., age of the person in the transport example. The conditional logit model assumes different values for each alternative and the impact of a unit of x_k is assumed to be constant across alternatives, e.g., journey time in the choice of transport mode. The conditional logit model was proposed into econometrics by McFadden (1974) and it has been used in biomedical research to estimate relative risks in matched case-control studies as well. Unfortunately, there is confusion in the literature for the terminology of the models. Some authors call `multinomial()` with (6.39) the "generalized logit model". Others call the mixed logit model the "multinomial logit model" and view the generalized logit and conditional logit models as special cases. In VGAM terminology there is no need to give different names to all these slightly differing special cases. They are all still called multinomial logit models, although it may be added that there are some covariate-specific linear/additive predictors. The important thing is that the framework accommodates $\boldsymbol{x}_{ij}$, so one tries to avoid making life unnecessarily complicated. And `xij` can apply in theory to any VGLM and not just to the multinomial logit model.

6.20.2 The `xij` Argument

VGAM handles variables whose values depend on η_j, (6.41), using the `xij` argument. It is assigned an S formula or a list of S formulas. Each formula, which must have M different terms, forms a matrix that premultiplies a constraint matrix. In detail, (6.39) can be written in vector form as

$$\boldsymbol{\eta}(\boldsymbol{x}_i) \;=\; \mathbf{B}^T \boldsymbol{x}_i \;=\; \sum_{k=1}^{p} \mathbf{H}_k \, \boldsymbol{\beta}_k^* \, x_{ik}, \tag{6.44}$$

where $\boldsymbol{\beta}_k^* = (\beta_{(1)k}^*, \ldots, \beta_{(r_k)k}^*)^T$ is to be estimated. This may be written

$$\boldsymbol{\eta}(\boldsymbol{x}_i) \;\;=\;\; \sum_{k=1}^{p} \mathrm{diag}(x_{ik}, \ldots, x_{ik}) \, \mathbf{H}_k \, \boldsymbol{\beta}_k^*. \tag{6.45}$$

To handle (6.40)–(6.41) we can generalize (6.45) to

$$\boldsymbol{\eta}_i \;\;=\;\; \sum_{k=1}^{p} \mathrm{diag}(x_{ik1}, \ldots, x_{ikM}) \, \mathbf{H}_k \, \boldsymbol{\beta}_k^* \quad (= \sum_{k=1}^{p} \mathbf{X}_{(ik)}^* \, \mathbf{H}_k \, \boldsymbol{\beta}_k^*, \text{ say}). \tag{6.46}$$

Each component of the list `xij` is a formula having M terms (ignoring the intercept) which specifies the successive diagonal elements of the matrix $\mathbf{X}_{(ik)}^*$. Thus each row of the constraint matrix may be multiplied by a different vector of values. The constraint matrices themselves are not affected by the `xij` argument.

How can one fit such models in VGAM? Here are the examples revisited.

1. ```
   fit1 = vglm(formula = cbind(leye,reye) ~ op,
               family = binom2.or(exchangeable=TRUE, zero=3),
               xij = list(op ~ lop + rop + fill1(lop)),
               form2 =  ~ op + lop + rop + fill1(lop), data=eyesdata)
   ```

   Here, `lop` and `rop` are the ocular pressures of the left and right eyes. The specific values of the vector `op` are not needed (unless plotted—see Section 6.20.5) because they are overwritten by `lop` and `rop` when forming $\mathbf{X}_{\text{VLM}}$. One could call `op` a "dummy" vector since its purpose is labelling, however this is not to be confused with dummy variables! The function `fill1()` makes the number of terms equal to three ($= M$ for `binom2.or()`) and the value it returns is a structure of zeros the same dimension as `lop`—here it is just a vector. One could have used `fill1(rop)` instead (or any vector really—why?). Each response term in the formulas in `xij` link with the same term in `formula`, so essentially it is for labelling purposes only (here, "`op`"). However, if `plotvgam()` is to be used on `fit1` then the terms in `formula` should match the first term of each (RHS) formula in `xij`—this could be achieved by replacing "`op`" by "`lop`". One can see this labelling by typing `model.matrix(fit1, type="vlm")` to get $\mathbf{X}_{\text{VLM}}$; there will be a column called "`op`". The argument `form2` contains *all* terms used; it creates an all-encompassing LM matrix from which columns are extracted out; this matrix is called $\mathbf{X}_{\text{F2}}$.

   By the way,

   ```
   bad2 = vglm(cbind(leye,reye) ~ lop + rop, data=eyesdata,
               fam = binom2.or(exchangeable=TRUE, zero=3))
   ```

   would result in the model

$$\begin{aligned} \text{logit}\, P(Y_{i1}=1) &= \beta^*_{(1)1} + \beta^*_{(1)2}\, x_{i21} + \beta^*_{(1)3}\, x_{i22} \\ \text{logit}\, P(Y_{i2}=1) &= \beta^*_{(1)1} + \beta^*_{(1)2}\, x_{i21} + \beta^*_{(1)3}\, x_{i22} \\ \log\, \psi &= \beta^*_{(2)1}, \end{aligned} \tag{6.47}$$

   which is inappropriate. Another related model is

$$\begin{aligned} \text{logit}\, P(Y_{i1}=1) &= \beta^*_{(1)1} + \beta^*_{(1)2} x_{i21} \\ \text{logit}\, P(Y_{i2}=1) &= \beta^*_{(1)1} + \beta^*_{(1)3} x_{i22} \\ \log\, \psi &= \beta^*_{(2)1}, \end{aligned}$$

   which can be fitted with

   ```
   cmat = matrix(c(1,1,0, 0,0,1), 3, 2)
   bad3 = vglm(cbind(leye,reye) ~ lop + rop,
               fam = binom2.or, data=eyesdata,
               constraints=list("(Intercept)"=cmat, lop=rbind(1,0,0), rop=rbind(0,1,0)))
   ```

   This is different to (6.42) because it allows for a different regression coefficient for each eye, i.e., the effect of ocular pressure on each eye is different. In other words, this model is not exchangeable. For this reason, it too is unsatisfactory.
   ```

2. Let's fit (6.43). Suppose the journey cost and time variables have had the cost and time of walking subtracted from them. Then, using "`.trn`" to denote train,

```
fit2 = vglm(cbind(bus,train,car,walk) ~ Cost + Time + Distance,
            fam = multinomial(parallel = FALSE ~ 1),
            xij = list(Cost ~ Cost.bus + Cost.trn + Cost.car,
                       Time ~ Time.bus + Time.trn + Time.car),
            form2 = ~  Cost.bus + Cost.trn + Cost.car +
                       Time.bus + Time.trn + Time.car +
                       Cost + Time + Distance,
            data=gotowork)
```

should do the job. It has $\mathbf{H}_1 = \mathbf{I}_3$ and $\mathbf{H}_2 = \mathbf{H}_3 = \mathbf{H}_4 = \mathbf{1}_3$ because the lack of parallelism only applies to the intercept. However, unless `Cost` is the same as `Cost.bus` and `Time` is the same as `Time.bus`, this model should not be plotted with `plotvgam()`; see Section 6.20.5 for details.

By the way, suppose $\beta^*_{(1)4}$ in (6.43) is replaced by $\beta^*_{(j)4}$. Then the above code but with

```
fam = multinomial(parallel = FALSE ~ 1 + Distance),
```

should fit this model. Equivalently,

```
fam = multinomial(parallel = TRUE ~ Cost + Time - 1),
```

As an exercise, discuss the relative merits of the two models, i.e., having $\beta^*_{(j)4}$ versus a single coefficient $\beta^*_{(1)4}$.

6.20.3 More Complicated Examples

The above examples are reasonably straightforward because the variables were entered linearly. However, things become more tricky if data-dependent functions are used in any `xij` terms, e.g., `bs()` or `poly()`. In particular, regression splines such as `bs()` can be used to estimate a general smooth function $f(x_{ij})$, which is very useful for exploratory data analysis.

For the `eyesdata` example the code

```
fitwrong = vglm(cbind(leye,reye) ~ bs(op), data=eyesdat,
                fam = binom2.or(exchangeable=TRUE, zero=3),
                xij = list(bs(op) ~ bs(lop) + bs(rop) + fill(bs(lop))),
                form2 =  ~ bs(op) + bs(lop) + bs(rop) + fill(bs(lop)))
```

is incorrect because the basis functions for `bs(lop)` and `bs(rop)` are not identical as the knots differ. Consequently, they represent two different functions despite having common regression coefficients.

Fortunately, it is possible to force the two `bs()` terms to have identical basis functions by using a trick: combine the vectors temporarily. To do this, one can use

```
BS = function(x, ..., df=3) bs(c(x,...), df=df)[1:length(x),,drop=FALSE]
```

This computes a B-spline evaluated at `x` but using other arguments as well to form an overall vector from which to obtain the (common) knots. Then the usage of `BS()` can be something like

```
fit5 = vglm(cbind(leye,reye) ~ BS(op),
            fam = binom2.or(exchangeable=TRUE, zero=3), data=eyesdat,
            xij = list(BS(op) ~ BS(lop,rop) + BS(rop,lop) + fill1(BS(lop,rop))),
            form2 = ~  BS(op) + BS(lop,rop) + BS(rop,lop) + fill1(BS(lop,rop)) +
                       op + lop + rop)
```

So `BS(lop,rop)` is the smooth term for `lop`, and `BS(rop,lop)` is the smooth term for `rop`. To plot the terms of `fit5` correctly, however, we can replace "`BS(op)`" by "`BS(lop,rop)`"—see Section 6.20.5 for details.

6.20.4 Prediction

The generic `predict()` should work as usual with `vglm()` models utilizing the `xij` argument provided the argument `newdata` is assigned a data frame with *all* the variables in the argument `form2`.

6.20.5 Plotting

Plotting via `plotvgam(vglmObject)` (where `vglmObject` uses the `xij` argument) requires some finesse. The details are as follows.

For a valid plot the important rules are:

(i) terms in RHS of `formula` should match both

(a) the LHS term (response) of each formula in the `xij` list, and

(b) the first term of the RHS of each formula in the `xij` list.

For example, `term1` and `term2` in

```
fit = vglm(response ~ term1 + term2 + term3 + ...,
           fam = VGAMfamilyfunction,
           xij = list(term1  ~ term1  + term1a + term1b + ...,
                      term2  ~ term2  + term2a + term3b + ...),
           form2 = ~  term1  + term2  + term3  + ... +
                      term1a + term1b + ... +
                      term2a + term2b + ... ,
           data=dataframe)
plotvgam(fit, se=TRUE)
```

Here, the first component functions of `term1` and `term2` should plot correctly against their first (inner) arguments, e.g., if `term1` is `myfun(x5,x6,df=4)` then its first inner argument is `x5`.

(ii) the `varxij`th (inner) argument of each such term is used for the plotting. The default is `varxij=1`, meaning the first. For example, suppose `term1` was `NS(dum1,dum2)`. It has two variables `dum1` and `dum2`, and so its component functions would be plotted against `dum1`. If `term2` was `NS(dum3,dum4)`. then its component functions would be plotted against `dum3` by default.

The above rules arise because the default for `plotvgam()` is `raw=TRUE`, meaning that if a constraint matrix $\mathbf{H}_k$ has r_k columns then $\mathbf{H}_k$ is temporarily replaced by

$$\mathbf{H}_k^* \;=\; \begin{pmatrix} \mathbf{I}_{r_k} \\ \mathbf{O} \end{pmatrix}, \tag{6.48}$$

so that

$$\mathbf{H}_k^*\, \boldsymbol{f}_k^*(x_k) \;=\; \begin{pmatrix} \boldsymbol{f}_k^*(x_k) \\ \mathbf{0} \end{pmatrix}.$$

Since only the first r_k component functions of x_k are plotted, these are then just the $\widehat{\boldsymbol{f}}_k^*(x_k)$ plotted against x_k.

The call `plotvgam(VGLMobject)` uses the `formula` argument of `VGLMobject` to obtain the x_k variable for which the plots of the component functions $\widehat{\boldsymbol{f}}_k^*(x_k)$ are produced. Since x_k may vary for each η_j this

means that only one of them is potentially correct. By default VGAM chooses the *first* argument (more generally the varxijth one) of the *first* term of the RHS of each formula in the xij list. For example, the term NS(dum1,dum2) has two variables dum1 and dum2, and so all the raw component functions for that term are plotted against dum1.

With terms affected by xij, the default values of some other arguments need to be changed to give a more accurate representation. For example, something like xlab="dum1" should be assigned in plotvgam() because NS(dum1,dum2) produces an xlab equalling dum1 and dum2 written on two lines. Also, it may be necessary to use the which.term and which.cf arguments to select only 'correct' component functions.

The above method of how plotvgam() works means that essentially only one-column constraint matrices are handled. If necessary use the which.cf argument to select the component function. Note that the xij argument is not restricted to one-column constraint matrices but plotting essentially is.

Example 1

The call

```
Fit1 = vglm(cbind(leye,reye) ~ BS(lop,rop),
            fam = binom2.or(exchangeable=TRUE, zero=3),
            xij = list(BS(lop,rop) ~ BS(lop,rop) + BS(rop,lop) + fill1(BS(lop))),
            form2 = ~ lop + rop +
                      BS(lop,rop) + BS(rop,lop) + fill1(BS(lop)),
            data=eyesdata)
plotvgam(Fit1, se=TRUE)
```

plots the estimated smooth component function against lop. To plot the (same) component function against rop try

```
plotvgam(Fit1, varxij=2, se=TRUE)
```

Example 2

As another example, suppose we wish to modify fit2 for plotting. This could be done with

```
Fit2 = vglm(cbind(bus,train,car,walk) ~ Cost.bus + Time.bus + Distance,
            fam = multinomial(parallel = FALSE ~ 1),
            xij = list(Cost.bus ~ Cost.bus + Cost.trn + Cost.car,
                       Time.bus ~ Time.bus + Time.trn + Time.car),
            form2 = ~ Cost.bus + Cost.trn + Cost.car +
                      Time.bus + Time.trn + Time.car +
                      Distance,
            data=gotowork)
plotvgam(Fit2, se=TRUE, lcol="red", scol="blue")
```

The downside of this is that "Cost.bus" is not a good name because it is not really only for bus. An alternative is to use

```
plotvgam(fit2, se=TRUE, lcol="red", scol="blue")
```

provided Cost is the same as Cost.bus and Time is the same as Time.bus.

Example 3

Here's another example. Suppose we wish to fit smooth functions in the `gotowork` data frame. Then

```
NS = function(x, ..., df=4) ns(c(x,...), df=df)[1:length(x),,drop=FALSE]

FIT2 = vglm(cbind(bus,train,car,walk) ~ NS(Cost.bus, Cost.trn, Cost.car) +
                                        NS(Time.bus, Time.trn, Time.car) +
                                        ns(Distance),
            fam = multinomial(parallel = FALSE ~ 1),
            xij = list(NS(Cost.bus, Cost.trn, Cost.car) ~
                       NS(Cost.bus, Cost.trn, Cost.car) +
                       NS(Cost.trn, Cost.car, Cost.bus) +
                       NS(Cost.car, Cost.bus, Cost.trn),
                       NS(Time.bus, Time.trn, Time.car) ~
                       NS(Time.bus, Time.trn, Time.car) +
                       NS(Time.trn, Time.car, Time.bus) +
                       NS(Time.car, Time.bus, Time.trn)),
            form2 = ~  NS(Cost.bus, Cost.trn, Cost.car) +
                       NS(Cost.trn, Cost.car, Cost.bus) +
                       NS(Cost.car, Cost.bus, Cost.trn) +
                       NS(Time.bus, Time.trn, Time.car) +
                       NS(Time.trn, Time.car, Time.bus) +
                       NS(Time.car, Time.bus, Time.trn) +
                       ns(Distance) +
                       Distance +
                       Cost.bus + Cost.trn + Cost.car +
                       Time.bus + Time.trn + Time.car,
            data=gotowork)
plotvgam(FIT2, se=TRUE, lcol="red", scol="blue")
```

should work. The fitted smooths are plotted against `Cost.bus` and `Time.bus`. Note that $\mathbf{H}_1 = \mathbf{I}_3$ and $\mathbf{H}_2 = \mathbf{H}_3 = \mathbf{H}_4 = \mathbf{1}_3$.

The call

```
plotvgam(FIT2, varxij=2)
```

plots the cost component function against `Cost.trn`, and the time component function against `Time.trn`.

Example 4

Consider the following code. Here, the constraint matrix for the `NS()` term has one column and there is an exchangeable error structure.

```
fit8 = vglm(cbind(nBnW,nBW,BnW,BW) ~ Age + NS(dum1,dum2),
            binom2.or(exchang=TRUE, zero=3),
            data = mydata,
            xij = list(NS(dum1,dum2) ~ NS(dum1,dum2) +
                                       NS(dum2,dum1) +
                                       fill1(NS(dum1))),
            form2 = ~  NS(dum1,dum2) + NS(dum2,dum1) + fill1(NS(dum1)) +
                       Age + dum3 + dum2 + dum1)
```

Then

```
par(mfrow=c(2,2))
mych = as.character(~ NS(dum1,dum2))[2]
plotvgam(fit8, which.term=mych, se=TRUE)
plotvgam(fit8, which.term=mych, se=TRUE, varxij=2)
```

confirms that adjustments are needed since two of the plots are plotted against the wrong predictor. They are also labelled incorrectly. One can select out and plot the ‘correct’ plots by

```
par(mfrow=c(1,2))
plotvgam(fit8, which.term=mych, se=TRUE,
         which.cf=1, xlab="dum1")
plotvgam(fit8, which.term=mych, se=TRUE,
         which.cf=2, xlab="dum2", varxij=2)
```

6.20.6 Last word

The `xij` argument operates *after* the ordinary $\mathbf{X}_{\text{VLM}}$ matrix is created. Then selected columns of $\mathbf{X}_{\text{VLM}}$ are modified from information in the constraint matrices, `xij` and `form2` arguments. That is, from $\mathbf{X}_{\text{F2}}$ and $\mathbf{H}_k$. This whole operation is possible because $\mathbf{X}_{\text{VLM}}$ remains structurally the same. The crucial equation is (6.46).

Other `xij` examples are given in the online help of `fill()` and `vglm.control()`.