package: amba 0.3.0

Additive Models for Business Applications

Charlotte Maia

June 13, 2010

This vignetted partially introduces the amba package, an R package for using additive models in business with an emphasis on functional estimates and missing values. The current package is not only incomplete, it's in a transitional state. There were some serious flaws in the original design and the current implementation is halfway through the process of correcting those flaws. The sections in this vignette are essentially from the previous version, updated slightly. A completely new vignette is planned for the next revision.

Implementation Notes

When fitting linear models, or more generally additive models, one tries to estimate "effects", that is, how some change in some explanatory variable influences the expected value of some response. In the earlier versions of this package, those effects were captured by "term" objects, environment-based objects following a class hierarchy. In the current version, term object have been replaced by "contribution" objects, which are function-based objects.

Consistent with the previous versions, contributions can be linear or smooth, and linear terms can be pure linear, categorical or polynomial.

One feature that's different here, from other's implementations, is that linear contributions may contain multiple parameters each. Another feature that's different, is that there's no overall intercept, and as a general rule, each contribution contains it's own intercept.

The top level algorithm, is designed to be indifferent to the class of contribution, and computes partial residuals in such a way that the models can produce good estimates even when there's a large amount of missing data.

Note, that the changes to the package have introduced some new problems. These should be fixed in the next revision.

Backfitting with Missing Explanatory Values

The author finds the idea of throwing away entire observations when one or two values are missing, somewhat barbaric. Here we present a simple solution (which is a natural extension term objects).

One way to think of residuals, is as some vector of values. If we start with the response values and subtract the overall mean, we get values with relatively high variance. If we then subtract the fitted values for the first term, the variance decreases. If we repeat for each term, the variance gradually decreases, until we are left with values with relatively low variance. In the ideal case, the residuals would have zero variance.

If we apply certain special conditions, then it is possible to only subtract a fitted value, where the corresponding explanatory value is valid (i.e. not missing). Where it is not valid, we just skip that subtraction operation (i.e. for that particular observation, the variance is not reduced as much). For this to work, each explanatory variable's partial residuals for each fit (not just the final fit) must be zero-centered. For smoothers this isn't a big issue, however conventional linear terms often do not satisfy

this zero-centered condition. Noting the centering condition applies to partial residuals in relation to an explanatory variable (not in relation to a parameter) and each explanatory may have multiple parameters associated with it. For our linear terms to satisfy it, we require extra parameters. Categorical terms require one parameter for each level, and polynomial terms, their own intercepts.

This produces overall residuals. We can produce partial residuals by adding a term's fitted values. If that particular term has missing values then the corresponding partial residuals will be invalid. However we still get valid partial residuals where other terms have missing values.

Note that we still require valid responses. Plus there are some issues with interactions which are still being explored. For implementation purposes we regard a numeric value as invalid if it is one of {NA, NaN, Inf, -Inf} and a factor as invalid if it is NA.

We could write standard residuals for an additive model as:

$$\begin{aligned} r_i &= y_i - \hat{\eta}_i \\ &= y_i - \left(\hat{\theta} + \sum_{\forall t} \hat{\mathrm{f}}_{[t][i]} \right) \end{aligned}$$

Where

y_i is the ith response value.

$\hat{\theta}$ is the overall intercept.

$\hat{\eta}_i$ is the ith overall fitted value.

r_i is the ith overall residual.

$\forall t$ means that we will sum over all terms.

$\hat{\mathrm{f}}_{[t][i]}$ is the ith fitted value for term t.

Standard partial residuals are achieved by merely by either excluding a particular term, or by adding a term's fitted values to the residuals above:

$$r^*_{[t^*][i]} = r_i + \hat{\mathrm{f}}_{[t^*][i]}$$

Where

t^* is a term, for which we are computing partial residuals.

$r^*_{[t^*][i]}$ is the ith partial residual for term t.

Achieving our overall residuals is trivial, we just modify the summation condition so that we only include valid values. Here we are making an assumption that invalid explanatory value results in an invalid fitted value.

$$r_i = y_i - \left(\hat{\theta} + \sum_{\forall t; \hat{\mathrm{f}}_{[t][i]} \in \mathbb{V}} \hat{\mathrm{f}}_{[t][i]} \right)$$

Where $\mathbb{V}$ indicates a valid number as described above. We achieve partial residuals using the same formula for standard partial residuals.

Example

Here we are going to use a made up dataset to demonstrate some of the things discussed so far. This dataset is pretty bad, and may be replaced in future versions of this package. The examples here a purely to demonstrate how to use the package, they are not intended to be "good" models. First we need load the packages and the data.

```
> library (amba)
> #may be changed...
> d = datafile ("amba", "sample", TRUE)
```

Here d represents a data.frame. A preview of the data.frame shows that it's a bit messy.

```
> d [1:10,]
     g1   g2      x1      x2      x3      x4        y
1     A    A      NA      NA      NA      NA 109.4480
2     B    C      NA -8.2178  3.6634      NA   1.3881
3     A <NA> -9.0099  0.0990  9.0099      NA  18.6786
4     A    B      NA      NA -4.6535      NA  -6.5811
5     B    D      NA  3.2673 -3.2673  4.8515 117.2306
6     A    B      NA -5.6436 -3.6634  6.2376 -50.2027
7  <NA>    B      NA -5.4455      NA      NA -45.0123
8     B    E      NA -7.2277  1.6832      NA  96.0798
9     B    F  3.4653  3.8614      NA  7.0297  78.0201
10    B    B      NA      NA -8.6139 -7.8218  37.8071
```

The last part of the preview output is information on the number of valid realisations. Note that there are only two complete realisations. We are only going to use four of the explanatories, so now we have five complete realisations.

```
> d [1:10,c (1:3, 6)]
     g1   g2      x1      x4
1     A    A      NA      NA
2     B    C      NA      NA
3     A <NA> -9.0099      NA
4     A    B      NA      NA
5     B    D      NA  4.8515
6     A    B      NA  6.2376
7  <NA>    B      NA      NA
8     B    E      NA      NA
9     B    F  3.4653  7.0297
10    B    B      NA -7.8218
```

We create terms using constructors, re-iterating the earlier point, in general there is one term to one variable.

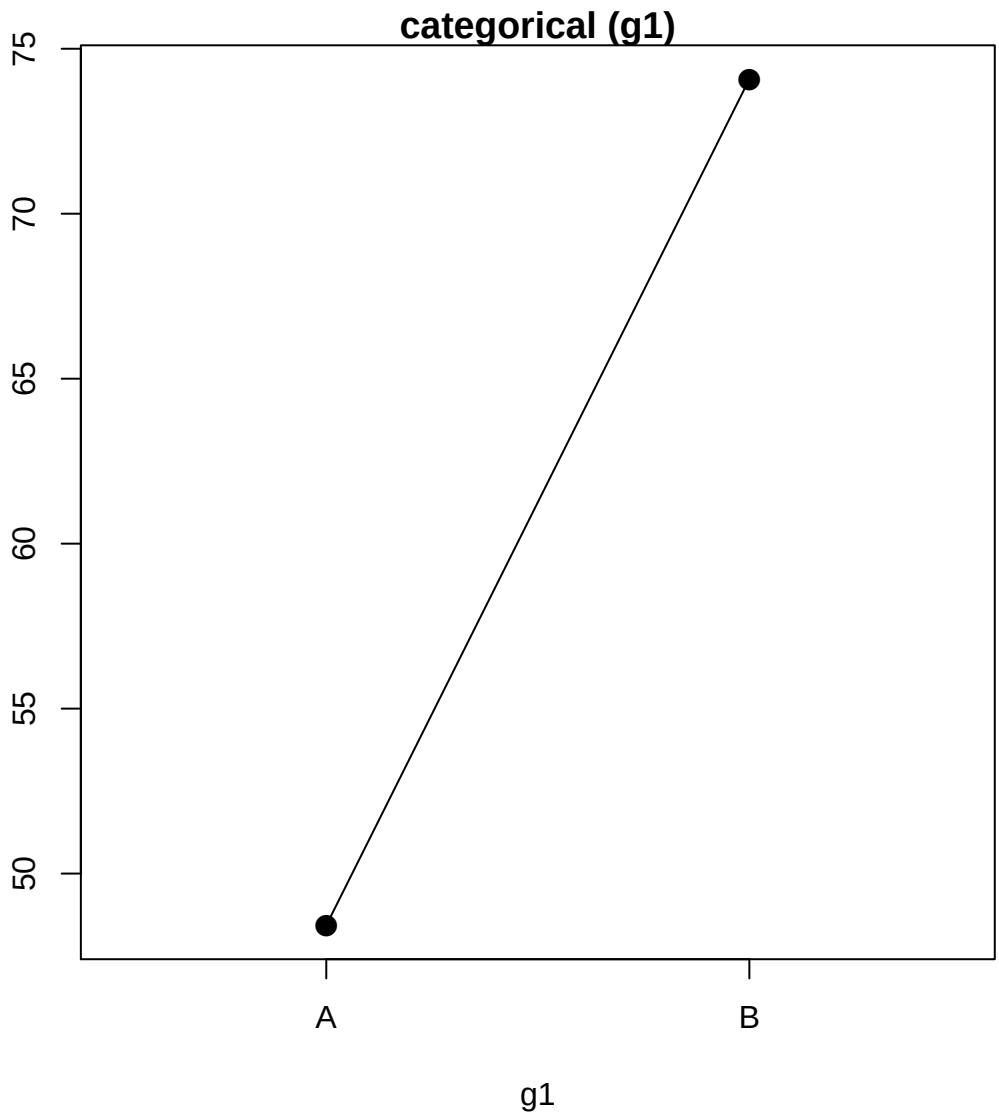
```
> t1 = categorical (g1)
> t2 = categorical (g2)
> t3 = linear (x1)
> t4 = linear (x4)
```

We can fit a term separately either by specifying the response as the second argument in the constructor, or by using the fit command. It is not necessary to do an explicit assignment. Terms are environments and the fit command will adjust the estimate object within the term. Functions summary and plot act as expected, except that the summary output is currently a mess and that a response (or partial residuals) are often required as an argument.

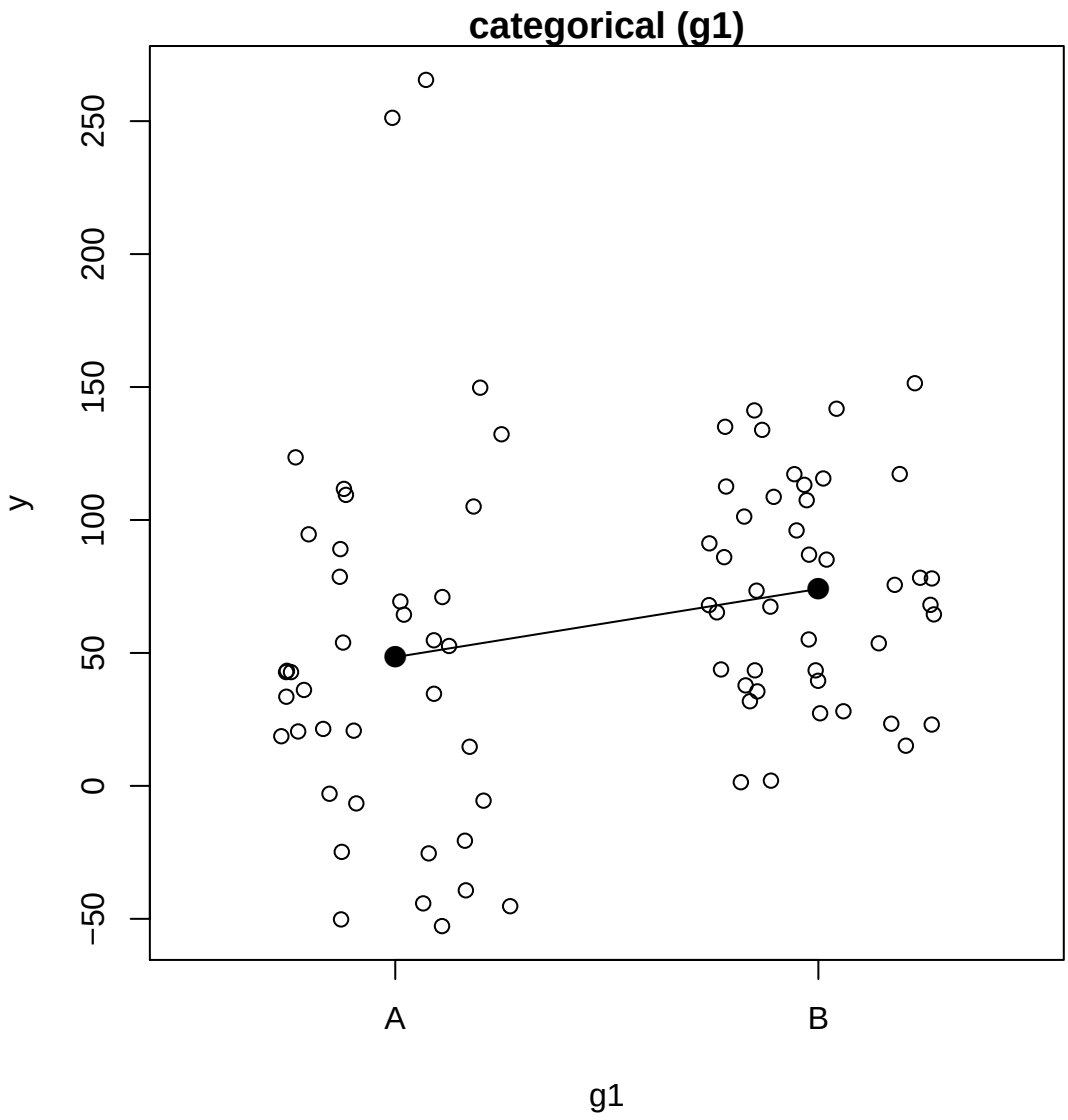
```
> fit (t1, y)
> summary (t1, y)
```

```
categorical (g1)
  parameter estimate
1         A 48.43023
2         B 74.07840
pcd: NA
```

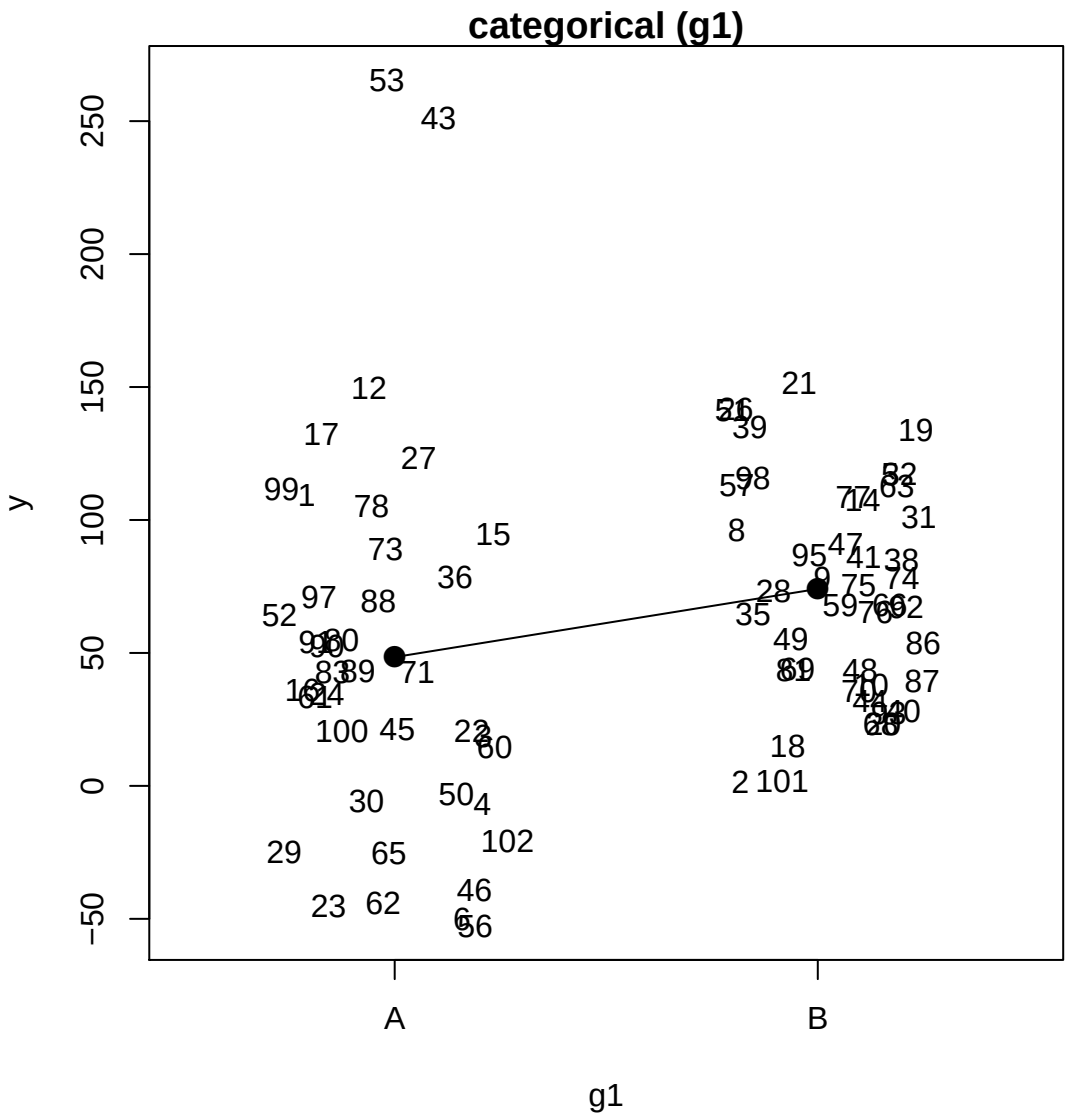
```
> plot (t1)
```




```
> plot (t1, y)
```



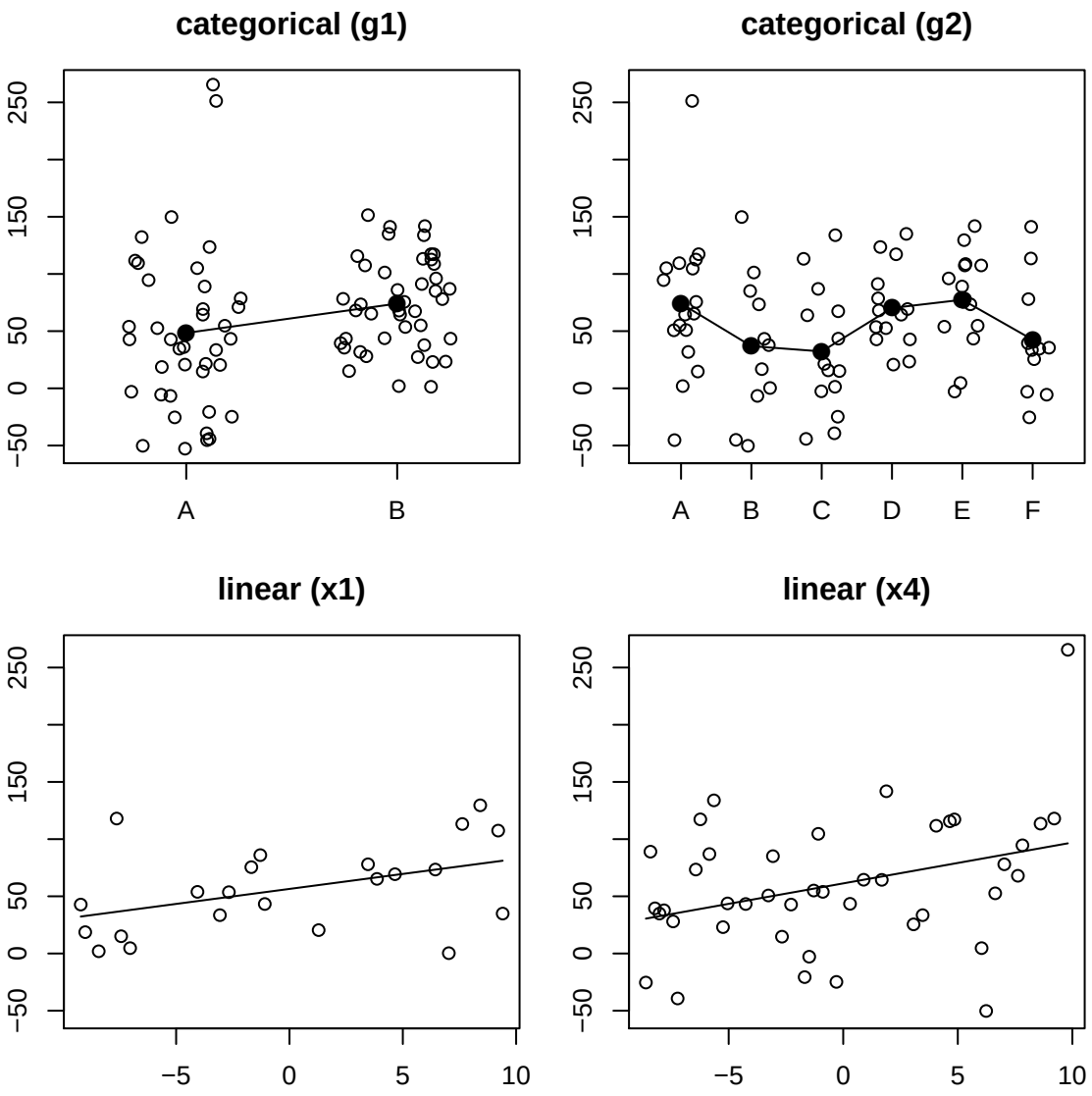

```
> plot (t1, y, index=TRUE)
```



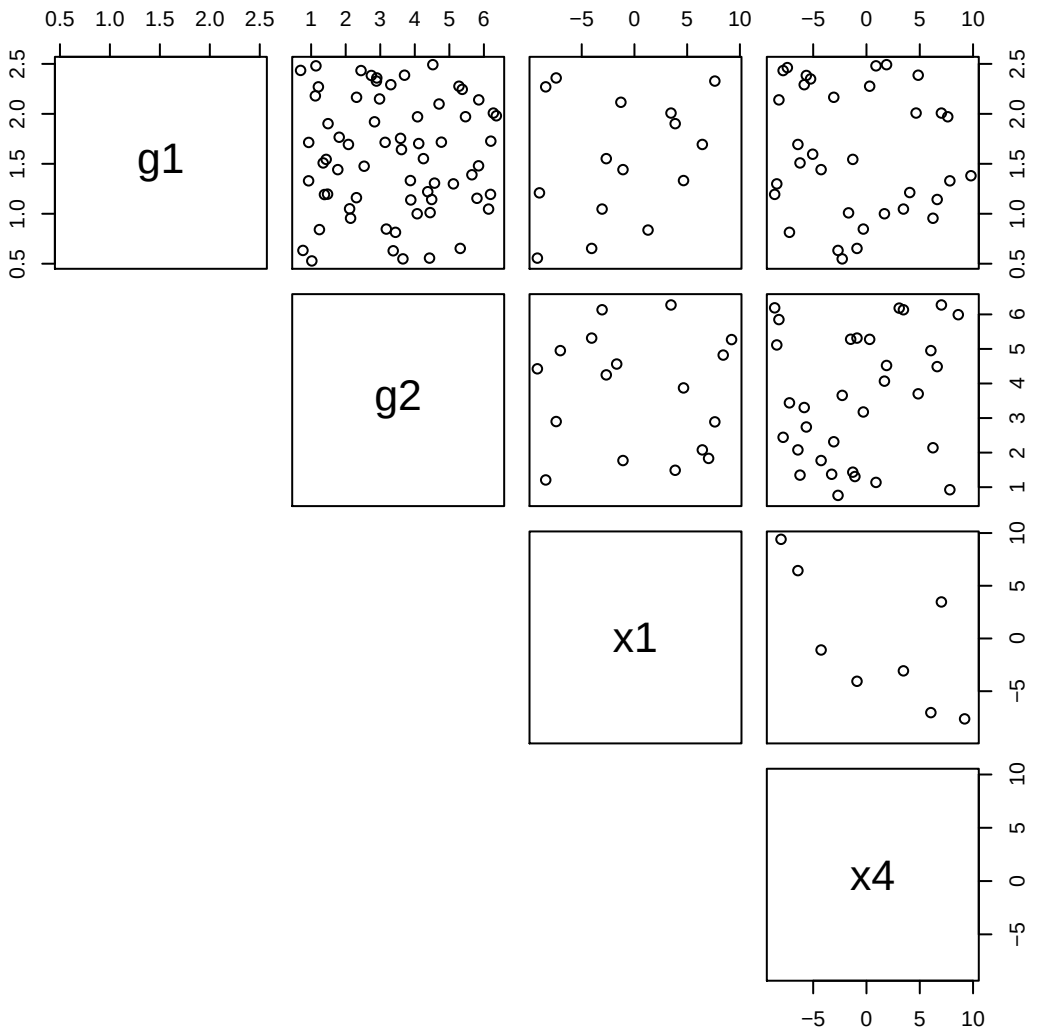
We can create a MMBA model, using the mmba function. Here the response is fit onto each term, ignoring the other terms. However, first we need to create a termlist object. Noting we can do both in one step if we want. It is also possible to plot the termlist object using the pairs function. We get something based on R's standard pairs plot, the main difference is that categorical variables are jittered.

```
> ts = t1 + t2 + t3 + t4
> m = mmba (y, ts)
```

```
> plot (m)
```


categorical (g1)
categorical (g2)
linear (x1)
linear (x4)
250
150
50
0
−50
A
B
C
D
E
F
−5
0
5
10

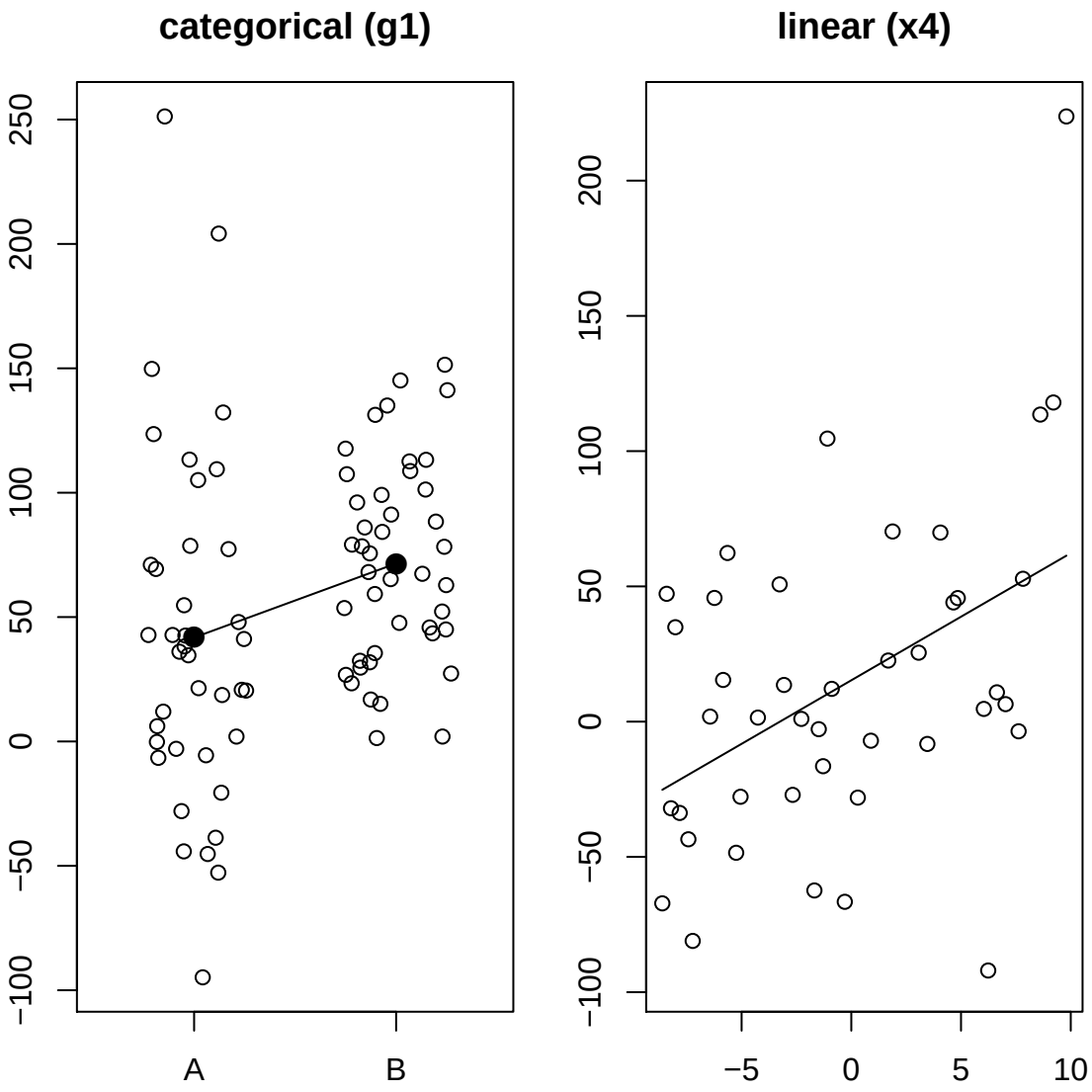
```
> pairs (ts)
```



We create an AMBA model basically the same, except using the amba function.

```
> m = amba (y, t1 + t4)
```

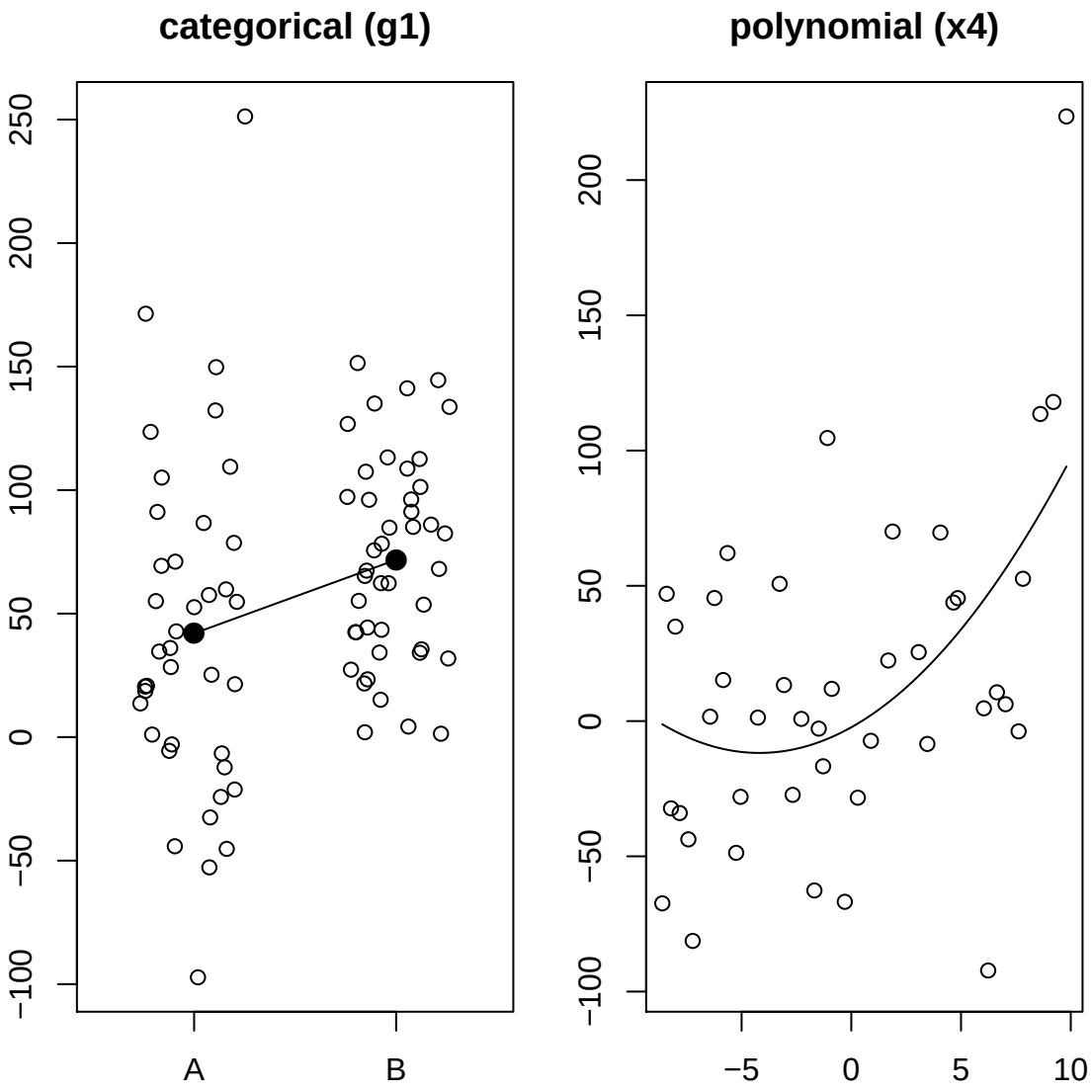
```
> plot (m)
```



We can can try a polynomial instead of a regular linear term.

```
> t4 = polynomial (x4, degree=2)
> m = amba (y, t1 + t4)
```

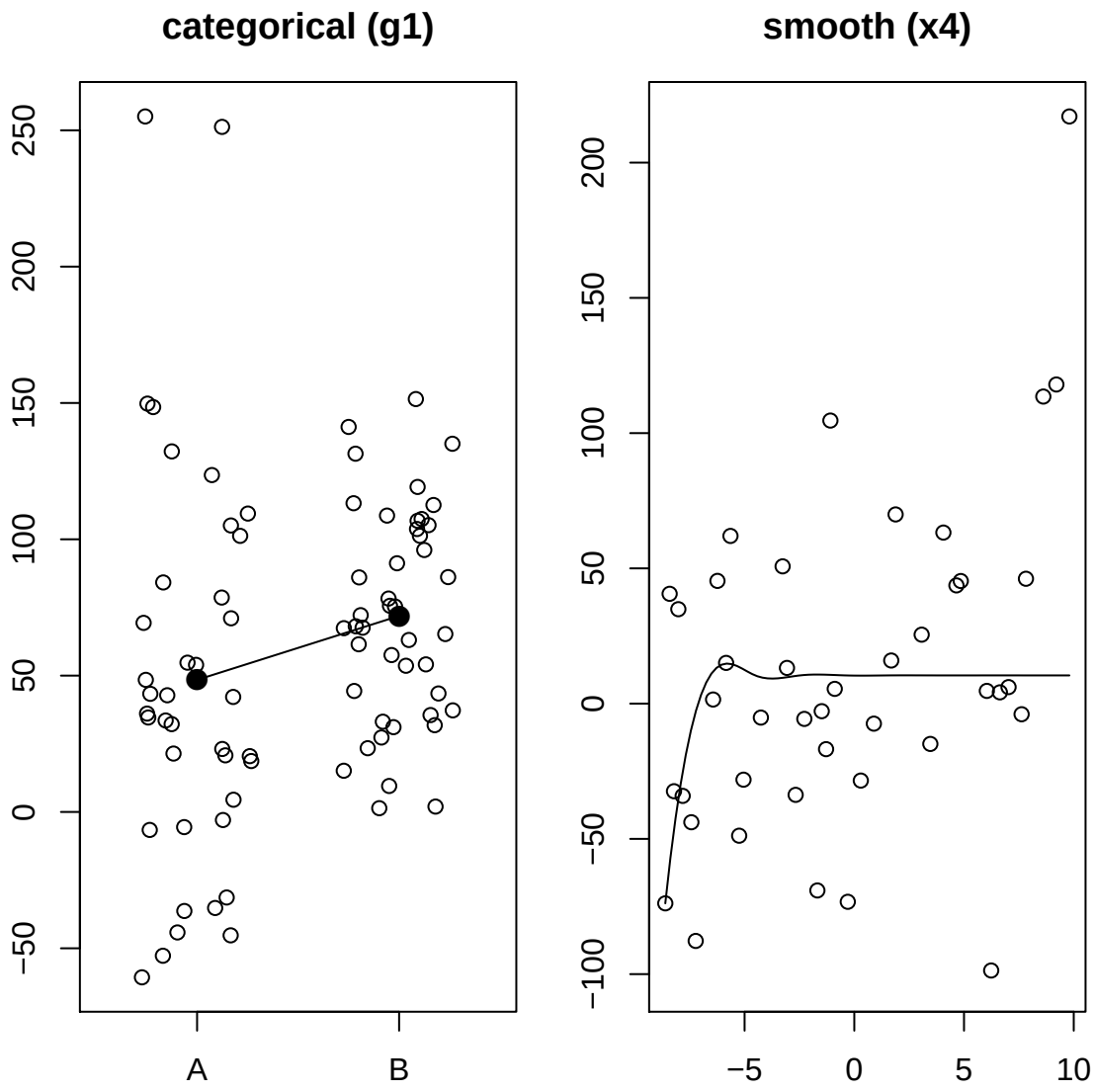
```
> plot (m)
```



Or create a semiparametric model using a smooth term.

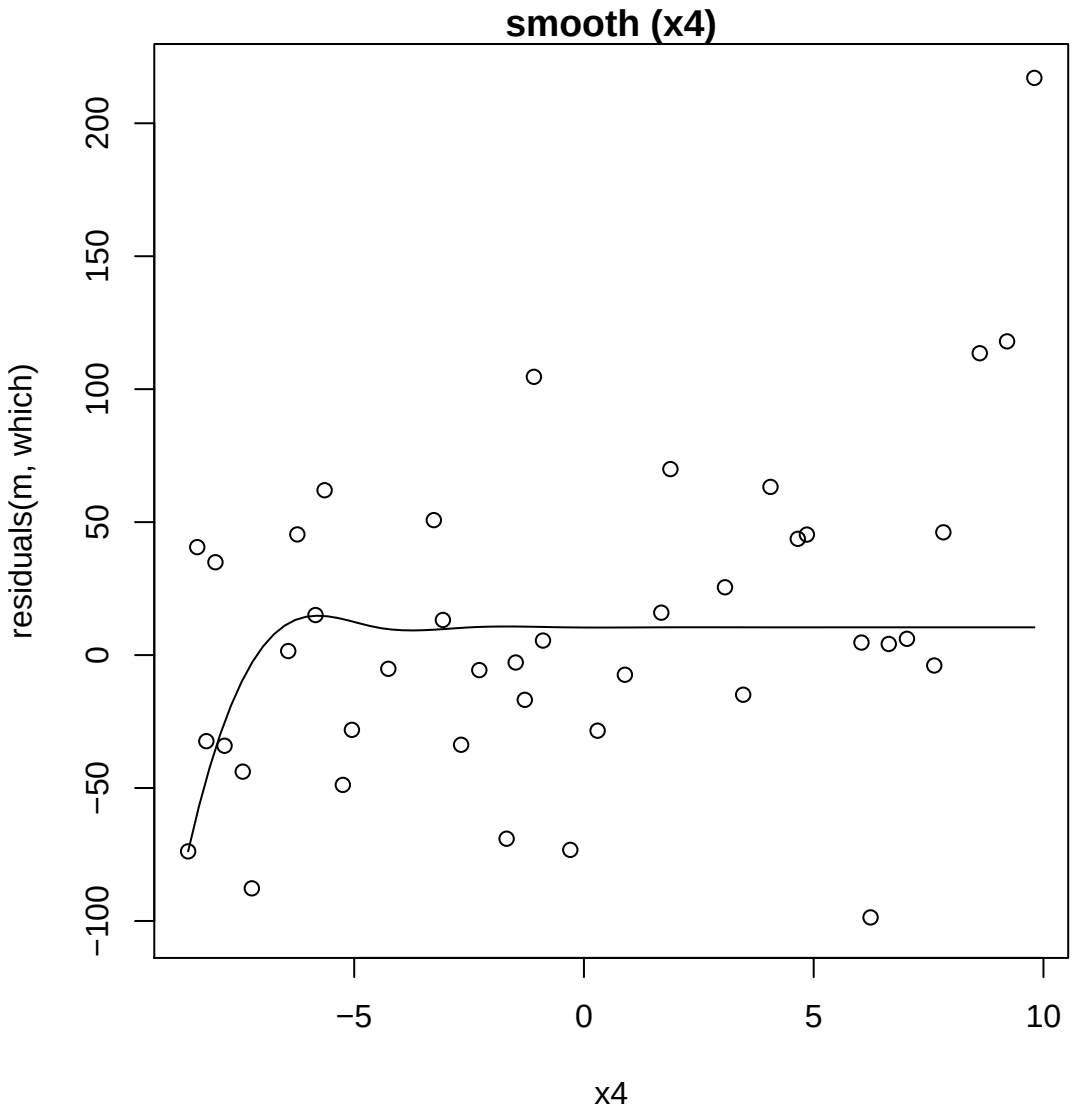
```
> t4 = smooth (x4, smoothness=0.9)
> m = amba (y, t1 + t4)


> plot (m)
```



Sometimes we just want to plot one of the terms from the AMBA model.

```
> plot (m, 2)
```



We can also get summary output for a single term, noting this uses the partial residuals, not the response itself. We can do the same thing (with some more work) by extracting the partial residuals and using the summary method for the term.

```
> summary (m, 2)
smooth (x4)
           sx        sy
1  -8.6139000 -73.83569
2  -6.5676889  10.43336
3  -4.5214778  10.43336
4  -2.4752667  10.43336
5  -0.4290556  10.43336
6   1.6171556  10.43336
7   3.6633667  10.43336
8   5.7095778  10.43336
9   7.7557889  10.43336
10  9.8020000  10.43336
```

```
pcd: NA
> summary (t4, residuals (m, 2))
smooth (x4)
           sx        sy
1  -8.6139000 -73.83569
2  -6.5676889  10.43336
3  -4.5214778  10.43336
4  -2.4752667  10.43336
5  -0.4290556  10.43336
6   1.6171556  10.43336
7   3.6633667  10.43336
8   5.7095778  10.43336
9   7.7557889  10.43336
10  9.8020000  10.43336
pcd: NA
```