Additive Models for Business Applications (Draft)

Charlotte Maia

November 19, 2009

Abstract

This vignette introduces the R package, amba, a currently incomplete package for using additive models in business. Here the main technical focus is on generalising the predictor (rather than generalising the response) by using term objects. Term objects are defined by an environment-based class hierarchy and include linear terms (which can contain multiple parameters each) and smooth terms (which are still being developed). Linear terms and smooth terms can be mixed to create semiparametric models. There is also a strong focus on interpretation (of effects), and on building models with missing explanatory values. There is very little support for statistical inference, and there are major restrictions on correlated explanatories.

Introduction

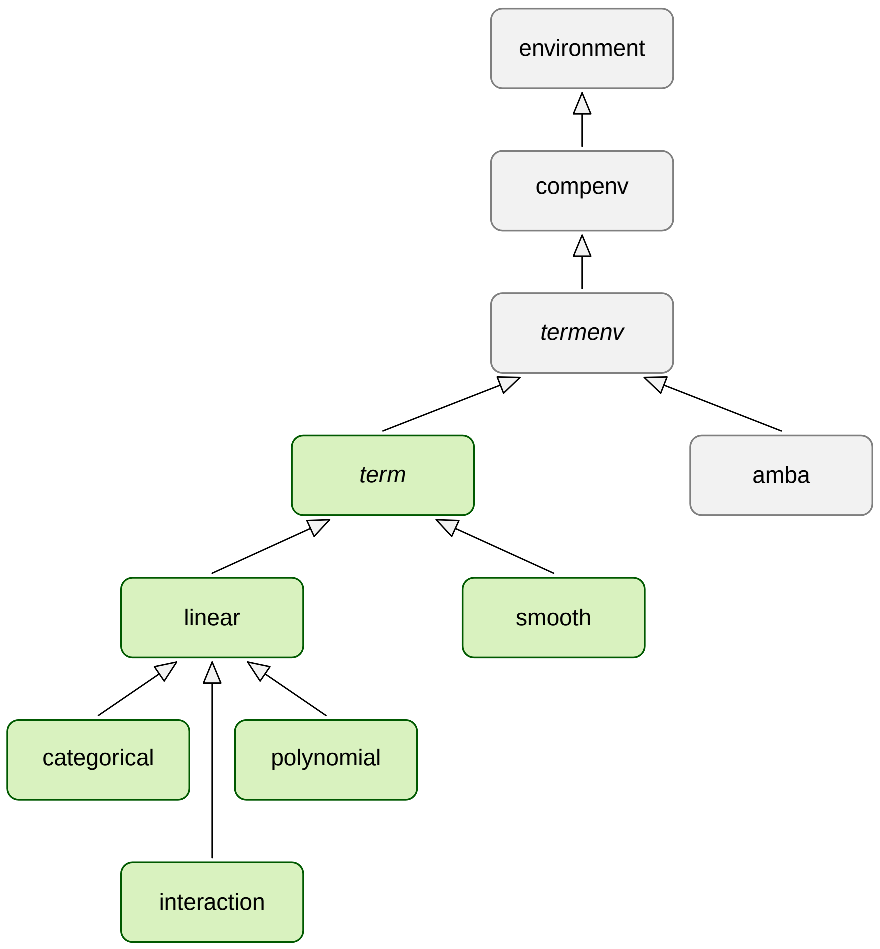
Here, we present a new kind of additive model, additive models for business applications (AMBAs). The package also implements simpler models, marginal models for business applications (MMBAs), however these are not discussed, except in the example section. The package is currently incomplete. It is of interest to use additive models in business, to model decision variables and produce historical/forecasting models, as well as more informal kinds of exploratory models. Hopefully, these features will be available soon.

The models are partly inspired by the works of Hastie and Tibshirani, as well as of Simon Wood, however have a different focus. Whilst their models tend to generalise the response side of things, and offer quite extensive support for statistical inference, here we restrict the response to the vaguely-normal case, restrict the use of correlated explanatory variables, as well as mostly doing away with statistical inference. On the other hand, there is a strong technical focus on generalising the predictor with term objects. The use of term objects opens up a number of possibilities.

Our additive predictor is essentially a list of such term objects. Each term object has a particular class, they can be categorical, smooth, etc (more on this later) and the user specifies the list of term objects when creating an AMBA model. Linear terms, are slightly unorthodox, in that each term may contain multiple parameter estimates. Smooth terms are based on local polynomial smoothing. Both smooth terms and interaction terms are still being developed. In general (the main exception being interactions), a term corresponds to a single explanatory variable, which is particularly useful for interpreting categorical and polynomial terms.

The models are fit using a backfitting algorithm, the algorithm is modified slightly, so that partial residuals are produced, even when there are missing explanatory values (more on this later too). Note that the response must be clean and the burden is on the user to ensure this. Except for correlated explanatories, we can generally mix and match linear and smooth terms however we want.

Note that the inference produced as part of the summary output is wrong, and may be removed entirely in future releases. Also note that in this vignette the words "realisation" and "observation" mean the same thing.



Term Objects

Term objects are used to represent both linear terms and smooth terms. They contain much of the information related to a term, design parameters, estimated parameters (or the estimated series), as well explanatory values. They are created using a constructor, fit using a set of response or partial residual values, and evaluated using a set of new possible explanatory values. It is possible to build a model using a single term, or an AMBA model using a list of terms.

Terms are defined by a class hierarchy and at present implemented (mostly) in S3. Roughly speaking, the root class is an abstract term class, however this ultimately extends environment, and there are a few things in between. An abstract term, is extended by a linear term and a smooth term, with a linear term (which here, isn't very user friendly except for very simple cases) being extended by a number of convenience classes. Subclasses of the smooth term are currently being developed.

One of the goals of the abstract term is to provide a data-structure that can be used to represent any kind of term, and then for the backfitting algorithm to be indifferent to the kind of term. For common functions such as plot or summary, they should behave in a very similar way for different kinds of term.

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. Also reiterating, the inference in the summary output is wrong and may be removed in the future. The examples here a purely to demonstrate how to use the package, they are not intended to be “good” models (they don’t even satisfy the convergence test, however this is only a draft). First we need load the packages and the data.

```
> library (oosp, warn=FALSE)
> library (amba, warn=FALSE)
> d = datafile ("amba", "sample", TRUE)
```

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

```
> preview (d)
factors:  g1 in {A, B}
          g2 in {A, B, C, D, E, F}
numerics: x1 in (-9.2079, 9.4059)
          x2 in (-10, 10)
          x3 in (-10, 10)
          x4 in (-8.6139, 9.802)
          y in (-52.7247, 265.5215)

data.frame <defective> ~ *(102, 7)
    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
100  A <NA>  1.2871  3.4653  1.8812      NA   20.4774
101  B    A -8.4158 -5.2475      NA      NA    1.9831
102  A <NA>      NA -2.8713 -6.8317 -1.6832 -20.6122

   g1 g2 x1 x2 x3 x4   y overall
nv 82 82 22 82 82 42 102       2
```

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.

```
> preview (d [,c (1:3, 6)])
factors:  g1 in {A, B}
          g2 in {A, B, C, D, E, F}
numerics: x1 in (-9.2079, 9.4059)
          x4 in (-8.6139, 9.802)

data.frame <defective> ~ *(102, 4)
    g1   g2      x1      x4
1    A    A      NA      NA
2    B    C      NA      NA
3    A <NA> -9.0099      NA
100  A <NA>  1.2871      NA
```

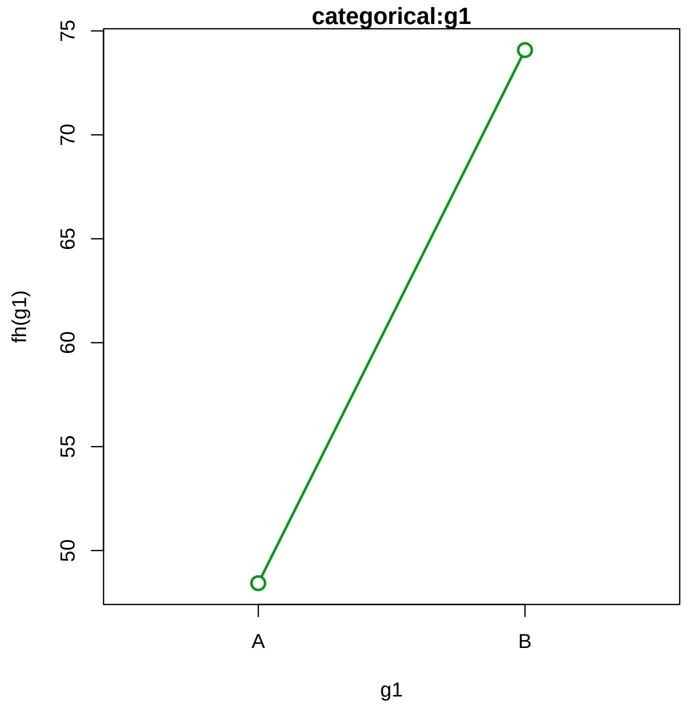
```
101  B    A -8.4158      NA
102  A <NA>      NA -1.6832

   g1 g2 x1 x4 overall
nv 82 82 22 42       5
```

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

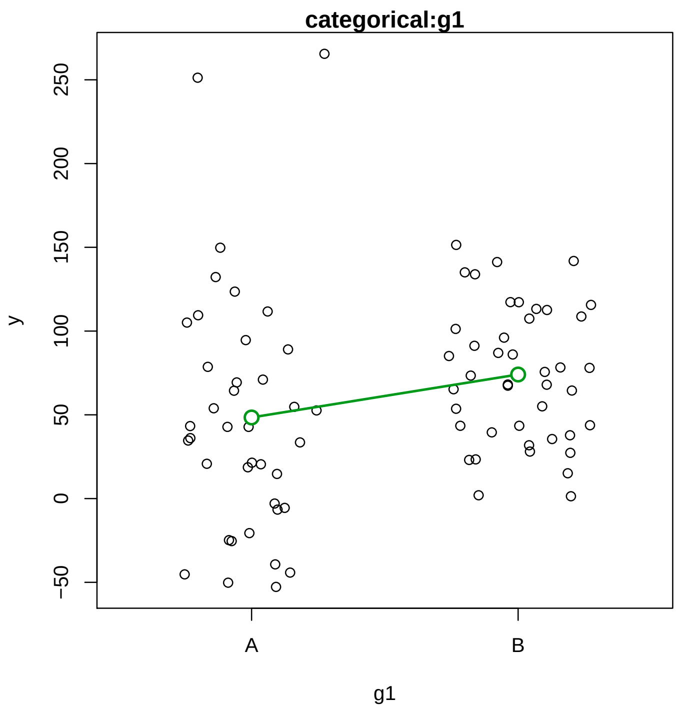
```
> t1 = categorical (g1)
> t2 = categorical (g2)
> t3 = linear (x1)
> t4 = linear (x4)
```

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.

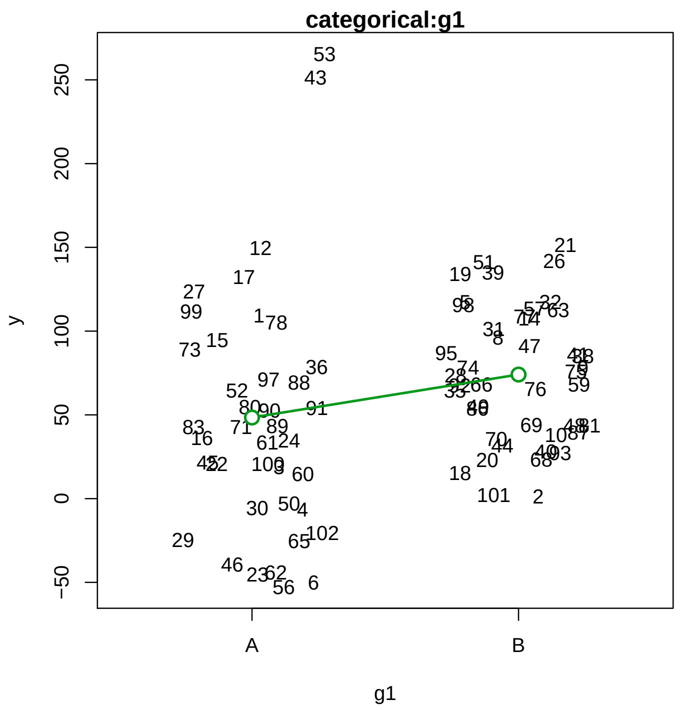
```
> fit (t1, y)
> summary (t1, y)
categorical:g1
  labs       th       se       tv           pv stars
1    A  0.00000 9.252821 5.234105 1.277431e-06   ***
2    B 25.64816 8.811954 8.406580 1.184386e-12  ****
  conditioning  nr(nv) eqnp        pcd
1         TRUE 102(82)    1 0.04738608


> plot (t1)
```



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



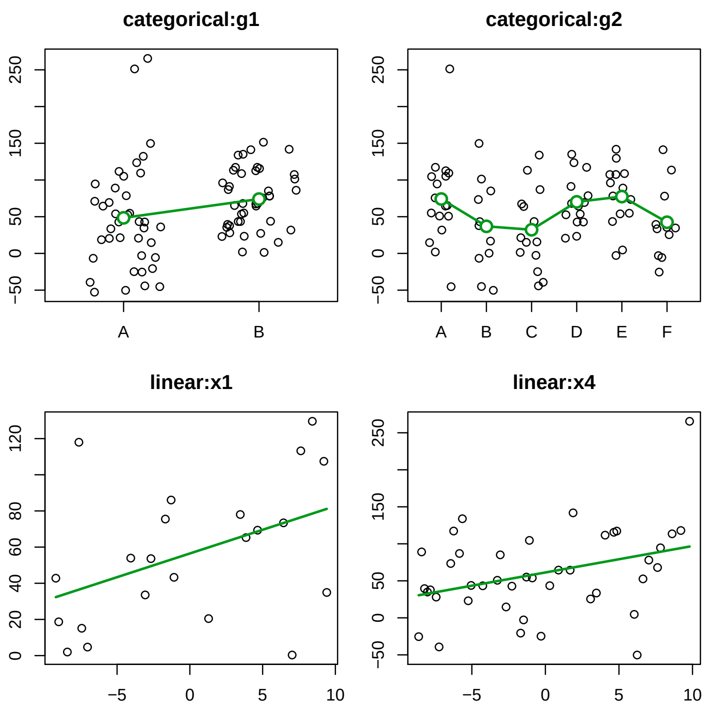
```
> plot (t1, y, index=TRUE)
```



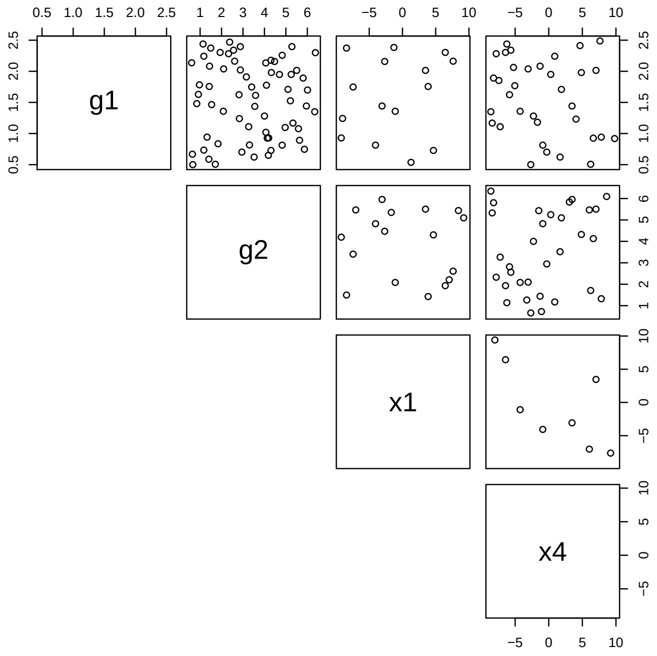
We can create a MMBA model, using the mmba function. Here the response is fit onto each term, ignoring the other terms. However, first we need to create a termlist object. Noting we can do both in one step if we want. It is also possible to plot the termlist object using the pairs function. We get something 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)
```

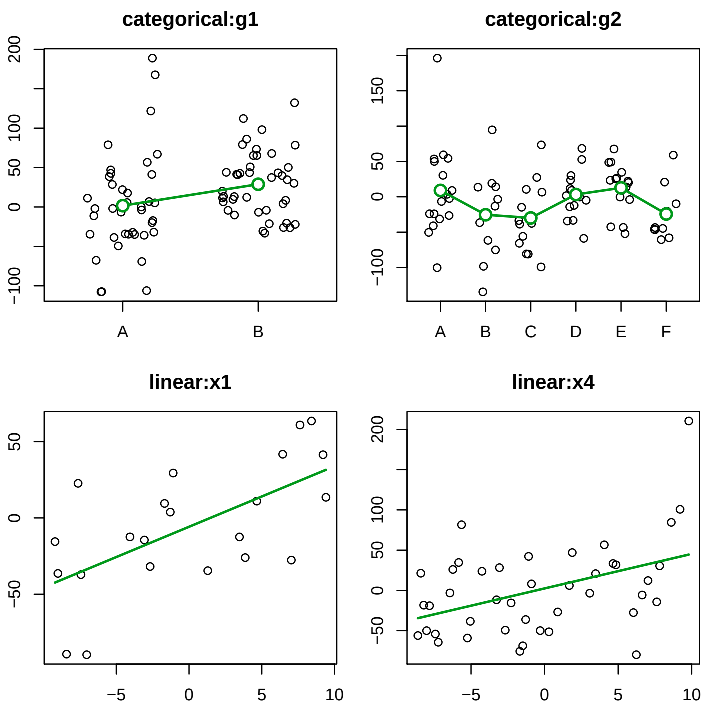


```
> pairs (ts)
```



We create an AMBA model basically the same, except using the amba function. Noting that the terms' estimates are reset before applying the backfitting algorithm.

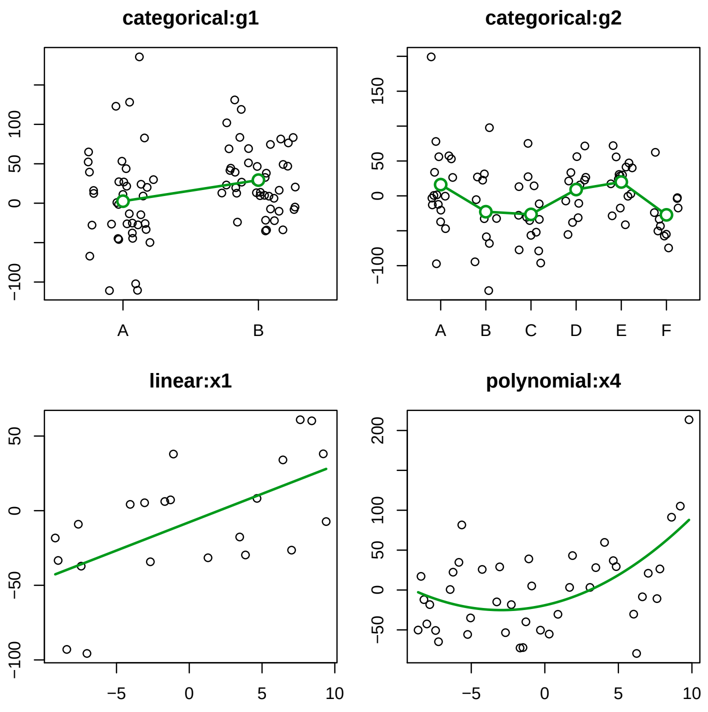
```
> m = amba (y, ts)
> summary (m)
intercept: 60.76099

categorical:g1
  labs       th       se        tv          pv stars
1    A  0.00000 8.913389 0.1910117 0.849046677
2    B 27.15447 8.488695 3.3994658 0.001097336    **
  conditioning  nr(nv) eqnp        pcd
1         TRUE 102(82)    1 0.06252509

categorical:g2
  labs         th       se         tv         pv stars
1    A   0.000000 12.17328  0.7447280 0.45882703
2    B -34.445862 15.13337 -1.6770939 0.09780147     .
3    C -38.968353 13.41430 -2.2291567 0.02888114     *
4    D  -5.779352 13.41430  0.2449944 0.80714813
5    E   3.603435 12.95944  0.9776049 0.33149843
6    F -33.408386 15.13337 -1.6085384 0.11203395
  conditioning  nr(nv) eqnp       pcd
1         TRUE 102(82)    5 0.1240363

linear:x1
  labs       th       se       tv         pv stars
1    x 3.971228 1.363373 2.912796 0.01210684     *
  conditioning  nr(nv) eqnp       pcd
1         TRUE 102(22)    1 0.3930225

linear:x4
  labs       th       se      tv          pv stars
1    x 4.290388 1.542856 2.78081 0.008889793    **
  conditioning  nr(nv) eqnp       pcd
1         TRUE 102(42)    1 0.1895406

  converged nfits(nsecs) nt(ncon) nr(ncomplete) eqnp overall.mar
1     FALSE       8(0.6)     4(4)        102(5)    9    37.37924


> plot (m)
```



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

```
> t4 = polynomial (x4, deg=2)
> m = amba (y, t1 + t2 + t3 + t4)
```

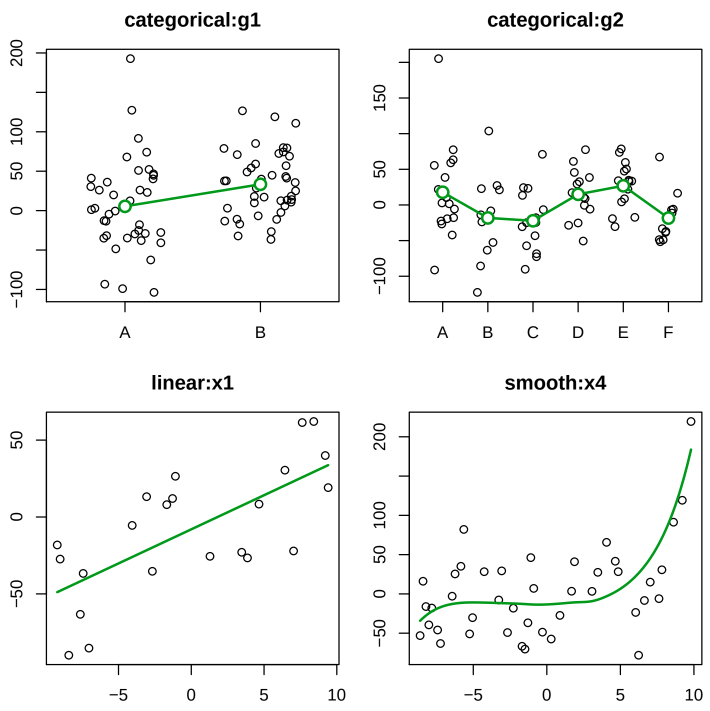
```
> plot (m)
```



Or create a semiparametric model using a smooth term.

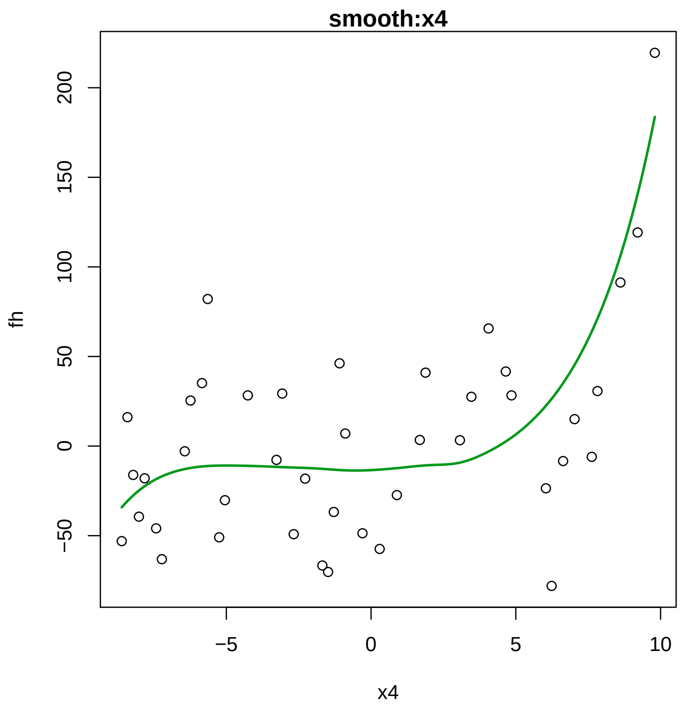
```
> t4 = smooth (x4)
> m = amba (y, t1 + t2 + t3 + t4)
```

```
> plot (m)
```



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

```
> plot (m, 4)
```



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, 4)
smooth:x4
  ns deg
1 20   2
sx:
 [1] -8.6139000 -7.6446421 -6.6753842 -5.7061263 -4.7368684 -3.7676105
 [7] -2.7983526 -1.8290947 -0.8598368  0.1094211  1.0786789  2.0479368
[13]  3.0171947  3.9864526  4.9557105  5.9249684  6.8942263  7.8634842
[19]  8.8327421  9.8020000
sy:
 [1] -34.201191 -20.479236 -13.753243 -11.205344 -10.884915 -11.294530
 [7] -11.911841 -12.554241 -13.561689 -13.323613 -12.011255 -10.624237
[13]  -9.408946  -3.620682   5.966046  20.382327  41.654688  72.716060
[19] 118.008408 183.721763
  conditioning  nr(nv) eqnp      pcd
1         TRUE 102(42)    4 0.629642
```

```
> summary (t4, residuals (m, 4))
smooth:x4
  ns deg
1 20   2
sx:
 [1] -8.6139000 -7.6446421 -6.6753842 -5.7061263 -4.7368684 -3.7676105
 [7] -2.7983526 -1.8290947 -0.8598368  0.1094211  1.0786789  2.0479368
[13]  3.0171947  3.9864526  4.9557105  5.9249684  6.8942263  7.8634842
[19]  8.8327421  9.8020000
sy:
 [1] -34.201191 -20.479236 -13.753243 -11.205344 -10.884915 -11.294530
 [7] -11.911841 -12.554241 -13.561689 -13.323613 -12.011255 -10.624237
[13]  -9.408946  -3.620682   5.966046  20.382327  41.654688  72.716060
[19] 118.008408 183.721763
  conditioning  nr(nv) eqnp      pcd
1         TRUE 102(42)    4 0.629642
```