

exploreGLM: Automatic exploration of a Generalized Linear Model.

Patricia Ballesta^a
patriciaballesta2@gmail.com

Jose Barrera-Gómez^{a,b}
jbarrera@creal.cat

October 6, 2013

^aDepartament de Matemàtiques, Universitat Autònoma de Barcelona (UAB), Bellaterra, Spain.

^bCentre for Research in Environmental Epidemiology (CREAL), Barcelona, Spain.

Contents

1	Introduction	2
2	Main functions of the <code>exploreGLM</code> package	2
2.1	Function <code>prepareData</code>	2
2.2	Function <code>exploreglm</code>	3
3	Illustrative examples of <code>exploreGLM</code>	4
3.1	Data <code>flowers</code>	4
3.2	Preparation and descriptive summary of the database <code>flowers</code>	5
3.3	Model fitting using <code>exploreglm</code>	7
3.3.1	Linear regression model	7
3.3.2	Logistic regression model	9
3.3.3	Poisson regression model	11
3.3.4	Logistic regression model for ordinal data	13
3.3.5	Logistic regression model for nominal data	14
	References	15

1 Introduction

This document was generated by R Sweave and it can be rescued through the instruction:

```
> vignette("exploreGLM")
```

The exploreGLM R package provides, on the one hand, an automatic descriptive summary of the data, taking into account the nature of each variable from the database (numerical or factor). On the other hand, it explores and automatically fits an appropriate Generalized Linear Model (GLM), according to the user's chosen response variable (by the argument `formula`). The regression models considered in this version are: linear, logistic, Poisson, and logistic regression model for an ordinal or nominal polytomous response variable.

The main functions of the exploreGLM package are `prepareData` and `exploreglm` which we describe in the next section.

2 Main functions of the exploreGLM package

In this section we will see how the user should use the exploreGLM package.

To start, we can load the library using:

```
> library("exploreGLM")
```

Information about the package is available using the help function:

```
> ?exploreGLM
```

2.1 Function prepareData

This function identifies and stores the information about the nature of each variable in the database. Then, the package can provide the appropriate descriptive summary for each variable. Furthermore, the function `prepareData` also prepares the database to make the detection of the appropriate GLM according to the type of response variable.

Information about the function usage is available using the help function:

```
> ?prepareData
```

The input arguments needed are the following:

- **data**: Data frame containing the variables to be described by `summary` and/or fitted by `exploreglm`.
- **k**: a numerical variables will be considered as categorical if its number of different values is not greater than k . Default value is $k = 5$.

- **id**: Name of the identifier variable, if any.

The variable associated to **id**, if any, will be omitted in the analyses.

2.2 Function `exploreglm`

This function fits the appropriate GLM according to the provided response variable in the **formula**. The class object `prepareData`, required as an input argument, informs about the response variable type and the number of different values that this variable presents. Once it is identified, and taking into account whether it is a numerical variable (indicated as **count** = TRUE/FALSE) or categorical (indicated as **ordinal** = TRUE/FALSE), the appropriate regression model will be applied amongst this five:

Response <i>Y</i>	Regression model	Function and Rlibrary
Continuous	Linear	<code>glm stats</code> family = gaussian
Dichotomous	Logistic	<code>glm stats</code> family = binomial
Count	Poisson	<code>glm stats</code> family = poisson
Categorical Ordinal	Polytomous ordinal	<code>polr MASS</code>
Categorical Nominal	Polytomous nominal	<code>multinom nnet</code>

Table 1: Types of models considered by the function `exploreglm`.

All remaining variables in the database, except the identifier variable (if any), are used as explanatory variables.

We can obtain the help of the function `exploreglm` by using the instruction:

```
> ?exploreglm
```

The input arguments needed are the following:

- **formula**: Formula that contains the variables to be explored.
- **object**: Object created by the function `prepareData`.
- **ordinal**: Indicated as TRUE if the response variable is ordinal. Default is FALSE.
- **count**: Indicated as TRUE if the response variable is a count. Default is FALSE.

Last two arguments (**count** and **ordinal**) require user's participation, because it is impossible for the computer to distinguish on its own between a numerical variable without decimals and a count, or between a nominal and a ordinal variable.

3 Illustrative examples of **exploreGLM**

3.1 Data flowers

To illustrate the package usage, we will use the database **flowers** included within the **exploreGLM** package. It can be loaded using:

```
> data(flowers)
```

This database was created from an own experiment with the aim to proof what was the effect produced by a drug substance diluted into the water that had been preserving a bouquet in a recipient for ten days. Clinical trial standards were applied to this experiment.

Data correspond to 120 flowers within three different classes: Dendranthema or white Chrysanthemum grandiflorum, red Chrysanthemum cinerariaefolium and yellow Dianthus caryophyllus, commonly known as Mum, Dalmatian and Carnation, respectively.

There were 8 different treatment combinations according to the type of the drug (Aspirin -ASA-, Ibuprofen, Paracetamol or Control -with no drug-) and the type of water (running or mineral water).

The help of the data **data(flowers)** can be obtained through instruction:

```
> ?flowers
```

The database variables are:

- **idflo**: An identifier for each flower.
- **treat**: Treatment given to the flower: **ASA**, **Control**, **Ibuprofen** or **Paracetamol**.
- **water**: Type of water: **Normal** or **Mineral**.
- **flower**: Flower species: **Carnation**, **Dalmatian** or **Mum**.
- **EVA**: A numeric variable of flower status at the end of follow-up. Scale 0-100 from little to very wilted.
- **color**: Water color at the end of follow-up: **Yellow** or **White**.
- **defects**: Defects count on the flower at the end of follow-up.
- **petal**: An ordered factor with levels: **Big** > **Medium** > **Small**.

3.2 Preparation and descriptive summary of the database flowers

Firstly, we apply the function `prepareData` to the database. If not specified, default value of k is 5:

```
> pDflo <- prepareData(data = flowers, id = idflo)
```

We can now see the prepared in a new format:

```
> pDflo
```

We can also explore only the first, say 18, rows of the prepared database:

```
> print(pDflo, lines = 18)
```

Variables (those with are least 6 different numerical values are assumed to be numerical):

	Name	Type	Different values
1	treat	categorical	4
2	water	categorical	2
3	flower	categorical	3
4	EVA	numerical	91
5	color	categorical	2
6	defects	numerical	24
7	petal	categorical	3

Printing the 18 first records:

	treat	water	flower	EVA	color	defects	petal
1	ASA	Normal	Dalmatian	71.5	Yellow	13	Big
2	ASA	Normal	Dalmatian	86.5	Yellow	24	Big
3	ASA	Normal	Dalmatian	87.0	Yellow	1	Small
4	ASA	Normal	Dalmatian	94.5	Yellow	14	Small
5	ASA	Normal	Dalmatian	92.5	Yellow	12	Big
6	ASA	Normal	Mum	89.5	Yellow	9	Medium
7	ASA	Normal	Mum	83.5	Yellow	3	Big
8	ASA	Normal	Mum	92.5	Yellow	8	Small
9	ASA	Normal	Mum	93.0	Yellow	17	Big
10	ASA	Normal	Mum	92.0	Yellow	8	Big
11	ASA	Normal	Carnation	50.5	Yellow	5	Small
12	ASA	Normal	Carnation	48.0	Yellow	1	Medium
13	ASA	Normal	Carnation	57.0	Yellow	5	Big
14	ASA	Normal	Carnation	70.0	Yellow	16	Big
15	ASA	Normal	Carnation	74.0	Yellow	2	Small
16	ASA	Mineral	Dalmatian	85.0	Yellow	18	Small
17	ASA	Mineral	Dalmatian	81.5	Yellow	17	Medium
18	ASA	Mineral	Dalmatian	92.0	Yellow	10	Small

Using the method `summary` we obtain the following descriptive summary:

```
> summary(pDflo, digits = 1)
```

```
=====
Number of Variables: 7      Observations: 120
Maximum number of categories in a numerical variable to be treated as
categorical: 5
=====

Qualitative Variables:
=====
Variable: treat , Missing values: 0 ( 0 %)
      ASA Control Ibuprofen Paracetamol Total
Frequencies  30      30      30      30  120
Percentages  25      25      25      25  100
-----
Variable: water , Missing values: 0 ( 0 %)
      Mineral Normal Total
Frequencies   60     60  120
Percentages   50     50  100
-----
Variable: flower , Missing values: 0 ( 0 %)
      Carnation Dalmatian Mum Total
Frequencies   40.0     40.0 40.0  120
Percentages   33.3     33.3 33.3  100
-----
Variable: color , Missing values: 0 ( 0 %)
      White Yellow Total
Frequencies   60     60  120
Percentages   50     50  100
-----
Variable: petal , Missing values: 0 ( 0 %)
      Big Medium Small Total
Frequencies  40.0   40.0 40.0  120
Percentages  33.3   33.3 33.3  100
-----

Quantitative Variables:
=====
      nMiss %Miss Mean  SD Min 2.5% 5% Q1 Median Q3 95% 97.5% Max
EVA      0      0 56.1 26.4 2.5 6 10.8 35.9 61.8 76.6 92.5 94.6 99.5
defects  0      0 12.5 7.0 1.0 1 2.0 6.8 12.5 18.2 23.0 24.0 24.0
```

In that descriptive summary, the qualitative variables firstly appear and then the quantitative ones. Notice too that the variable *idflo*, known as an identifier, does not appear in the summary and it will not be used either in model fitting.

3.3 Model fitting using `exploreglm`

3.3.1 Linear regression model

Let's consider the case in which we want to evaluate *EVA* as the response variable. This variable is continuous and for these type of variables the function `exploreglm` fits a linear regression model, so no values will be needed for arguments `ordinal` or `count`. Using `'~.'` in the formula, all explanatory variables will be considered.

```
> modEVA <- exploreglm(EVA ~ ., object = pDflo)
```

Through method `print` created for the objects of class `exploreglm`, we will obtain some basic information about our dependent variable, the type of fitted model and its associated formula.

```
> modEVA
```

```
=====
Dependent variable: EVA (Continuous variable)
Model type: Linear regression model
Model: EVA ~ treat + water + flower + color + defects + petal
=====
Further information about this model with summary()
```

To access the information about coefficients and model adjustments, we will use the method `summary`, that adds the previous head to the generic method `summary` for `glm`.

```
> summary(modEVA)
```

```
=====
Dependent variable: EVA (Continuous variable)
Model type: Linear regression model
Model: EVA ~ treat + water + flower + color + defects + petal
=====
```

Call:

```
glm(formula = formula, family = gaussian, data = data)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-49.18	-11.71	1.30	13.35	38.83

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	106.0062	7.7265	13.720	< 2e-16 ***
treatControl	-46.0740	6.2711	-7.347	3.92e-11 ***

treatIbuprofen	-47.1996	6.2790	-7.517	1.67e-11	***
treatParacetamol	-23.7886	5.0973	-4.667	8.75e-06	***
waterNormal	-8.4458	3.7555	-2.249	0.0265	*
flowerDalmatian	-11.4732	4.8108	-2.385	0.0188	*
flowerMum	-8.2738	4.3187	-1.916	0.0580	.
colorYellow	-6.8823	5.5592	-1.238	0.2184	
defects	-0.4073	0.2695	-1.511	0.1336	
petalMedium	3.2545	4.5247	0.719	0.4735	
petalSmall	-7.1146	4.4800	-1.588	0.1152	

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

(Dispersion parameter for gaussian family taken to be 368.7504)

Null deviance: 83179 on 119 degrees of freedom
 Residual deviance: 40194 on 109 degrees of freedom
 AIC: 1062.2

Number of Fisher Scoring iterations: 2

Generic graphs of the diagnostic model, can be obtained through instruction `plot(modEVA)`.

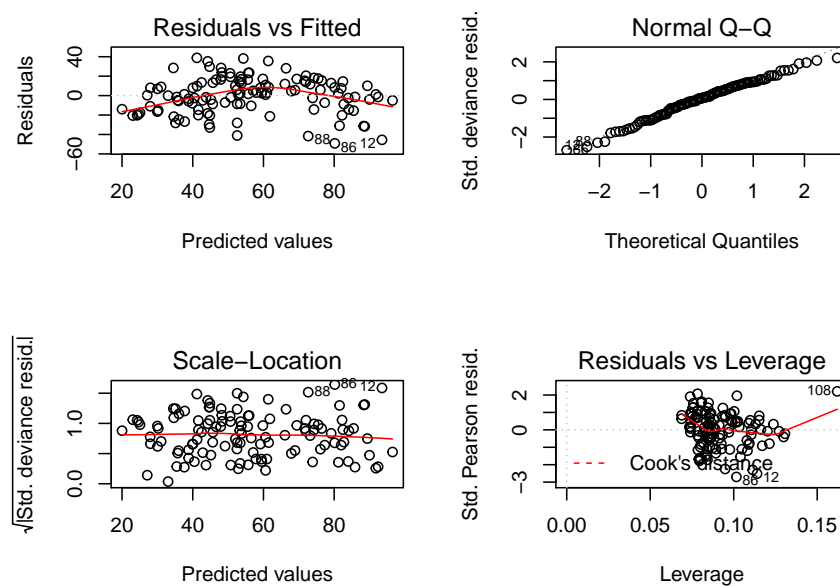


Figure 1: Graphical diagnosis of the linear model for the response *EVA*.

3.3.2 Logistic regression model

Should we want to evaluate the *color* as the response variable, which is dichotomous, function `exploreglm` will apply a logistic regression model as a function of some variables in the database (for example: *treat*, *water*, *flower* and *petal*).

```
> modcol <- exploreglm(color ~ treat + water + flower + petal,
+                       object = pDflo)
> modcol
```

```
=====
Dependent variable: color (Dichotomous variable)
Model type: Logistic regression model
Model: color ~ treat + water + flower + petal
=====
```

Further information about this model with `summary()`

The summary of the model is obtained as usually:

```
> summary(modcol)
```

```
=====
Dependent variable: color (Dichotomous variable)
Model type: Logistic regression model
Model: color ~ treat + water + flower + petal
=====
```

Call:

```
glm(formula = formula, family = binomial, data = data)
```

Deviance Residuals:

	Min	1Q	Median	3Q	Max
	-1.79259	-0.00007	-0.00001	0.51399	1.89869

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.15231	0.96287	1.197	0.2314
treatControl	-22.00588	2085.98343	-0.011	0.9916
treatIbuprofen	-21.86101	2085.98337	-0.010	0.9916
treatParacetamol	0.23056	0.81209	0.284	0.7765
waterNormal	1.67064	0.66106	2.527	0.0115 *
flowerDalmatian	20.53388	2085.98331	0.010	0.9921
flowerMum	0.08846	0.80829	0.109	0.9129
petalMedium	-1.30264	0.89616	-1.454	0.1461
petalSmall	-0.86546	0.83468	-1.037	0.2998

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

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 166.355 on 119 degrees of freedom
 Residual deviance: 62.028 on 111 degrees of freedom
 AIC: 80.028

Number of Fisher Scoring iterations: 19

Similarly, we can get graphical information using `plot(modcol)`.

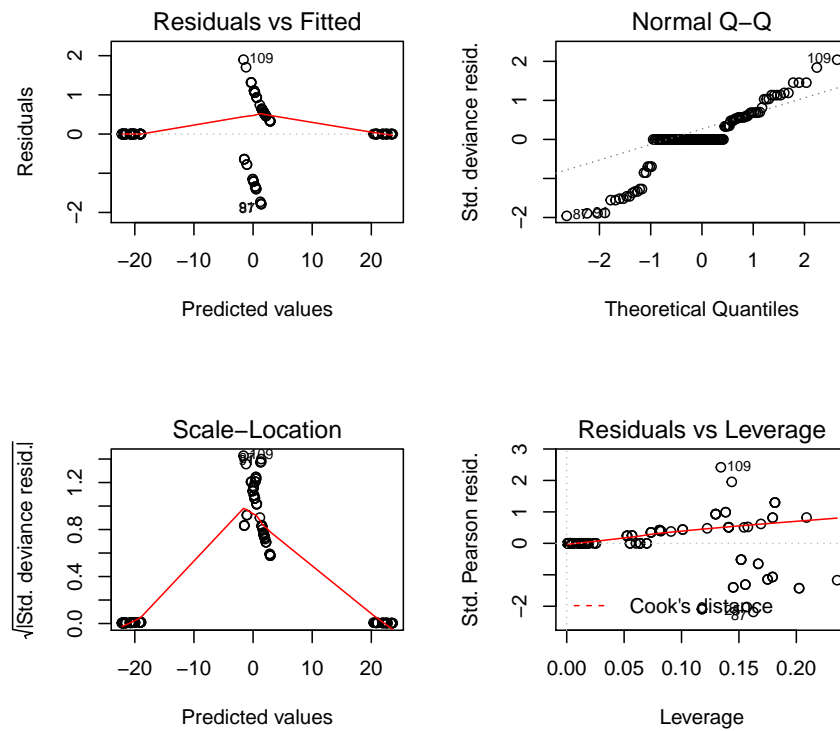


Figure 2: Graphical diagnosis of the logistic regression model for the response *color*.

3.3.3 Poisson regression model

If the response variable is *defects*, which is a count, the function `exploglm` fits a Poisson regression model. In that case, we should indicate `count = TRUE`. Variables selected for the model could be for example: *treat*, *water*, *flower* and *petal*. We can use `*` or `:` in the formula to compute some interactions.

```
> moddefects <- exploglm(defects ~ treat * flower + water + petal,
+                          object = pDflo, count = TRUE)
> moddefects
```

```
=====
Dependent variable: defects (Count variable)
Model type: Poisson regression model
Model: defects ~ treat + flower + water + petal + treat:flower
=====
```

Further information about this model with `summary()`

The summary of the model is provided as:

```
> summary(moddefects)
```

```
=====
Dependent variable: defects (Count variable)
Model type: Poisson regression model
Model: defects ~ treat + flower + water + petal + treat:flower
=====
```

Call:

```
glm(formula = formula, family = poisson, data = data)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-5.399	-1.540	-0.024	1.282	3.635

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.61062	0.10917	23.914	< 2e-16 ***
treatControl	0.30926	0.12848	2.407	0.0161 *
treatIbuprofen	-0.09860	0.13780	-0.716	0.4743
treatParacetamol	-0.03805	0.13651	-0.279	0.7805
flowerDalmatian	0.16641	0.12902	1.290	0.1971
flowerMum	0.02995	0.13332	0.225	0.8223
waterNormal	-0.23435	0.05318	-4.407	1.05e-05 ***
petalMedium	-0.02930	0.06648	-0.441	0.6594

petalSmall	-0.14953	0.06809	-2.196	0.0281 *
treatControl:flowerDalmatian	-0.16905	0.17282	-0.978	0.3280
treatIbuprofen:flowerDalmatian	0.30172	0.17916	1.684	0.0922 .
treatParacetamol:flowerDalmatian	-0.29937	0.19172	-1.562	0.1184
treatControl:flowerMum	-0.35412	0.18418	-1.923	0.0545 .
treatIbuprofen:flowerMum	0.11259	0.18810	0.599	0.5494
treatParacetamol:flowerMum	-0.07043	0.18986	-0.371	0.7107

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

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 522.11 on 119 degrees of freedom
 Residual deviance: 451.19 on 105 degrees of freedom
 AIC: 978.78

Number of Fisher Scoring iterations: 5

Also in this case we can get a graphical diagnosis of the model:

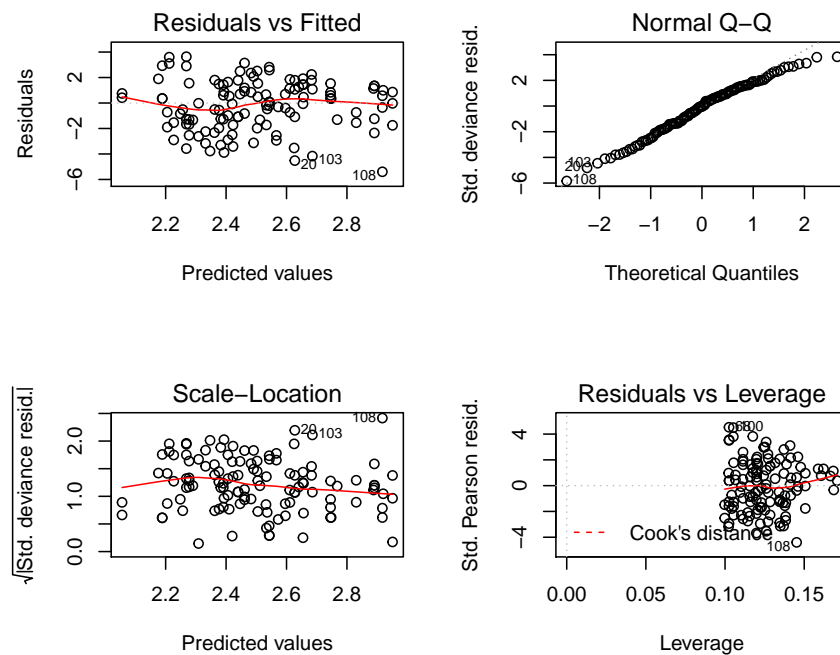


Figure 3: Graphical diagnosis of the Poisson regression model for the response *defects*.

3.3.4 Logistic regression model for ordinal data

In the next example, the response variable that we want to evaluate is *petal*, which is ordinal, so the function `exploreglm` will apply an ordinal regression model. In this case `ordinal = TRUE` should be indicated and the variables selected for the adjustment will be for example: *treat*, *water* and *flower*.

```
> modpetal <- exploreglm(petal ~ treat + water + flower,  
+                        object = pDflo, ordinal = TRUE)
```

The summary is:

```
> summary(modpetal)
```

```
=====
```

Dependent variable: petal (Ordinal variable)

Model type: Logistic regression model for polytomous ordinal data

Model: petal ~ treat + water + flower

```
=====
```

Call:

```
polr(formula = formula, data = data)
```

Coefficients:

treatControl	treatIbuprofen	treatParacetamol	waterNormal
-0.6009088	-0.8916779	-0.1846732	-0.7138493
flowerDalmatian	flowerMum		
-0.2641179	-0.1045554		

Intercepts:

Big Medium	Medium Small
-1.6343059	-0.1664863

Residual Deviance: 255.1101

AIC: 271.1101

As no graphs were implemented in this case, the method `plot` will return this message:

```
> plot(modpetal)  
> 'Error in plot.exploreglm(modpetal):  
+   ordinal and nominal models does not have a defined plot in  
+   exploreGLM package'
```

3.3.5 Logistic regression model for nominal data

Let's consider the case in which we want to evaluate *treat* as the response variable. This variable is nominal and for these type of variables the function `exploreglm` fits the regression model for nominal data, so no arguments will be needed in `ordinal` and `count`. Using '`~ .`' in the formula, `exploreglm` will apply a nominal regression model as a function of the remaining variables in the database.

```
> modtreat <- exploreglm(treat ~ ., object = pDflo)
```

The summary of the model gives:

```
> summary(modtreat)
```

```
=====
Dependent variable: treat (Nominal variable)
Model type: Logistic regression model for polytomous nominal data
Model: treat ~ water + flower + EVA + color + defects + petal
=====
```

```
Call:
multinom(formula = formula, data = data)
```

Coefficients:

	(Intercept)	waterNormal	flowerDalmatian	flowerMum	EVA
Control	10.243513	0.06696616	15.156248	0.1267937	-0.12899709
Ibuprofen	11.390908	-0.16736511	15.297918	0.0420634	-0.13364730
Paracetamol	6.802391	-0.65253290	0.412417	0.2563469	-0.08798593

	colorYellow	defects	petalMedium	petalSmall
Control	-18.4764507	0.01816238	0.6943748	-1.3457637
Ibuprofen	-18.7944792	-0.01655038	0.1830185	-1.9025946
Paracetamol	-0.1673726	-0.05214193	1.6237203	-0.1151715

Residual Deviance: 193.7425

AIC: 247.7425

As no graphs were implemented in this case, the method `plot` will return this message:

```
> plot(modtreat)
> 'Error in plot.exploreglm(modtreat) :
+ ordinal and nominal models does not have a defined plot in
+ exploreGLM package'
```

Note: Although the models doesn't have a completely practical sense, they serve to illustrate the use of the `exploreglm` function.

References

- [1] Ballesta P, Barrera-Gómez J. *R Package exploreGLM*. Version 0.2. 2013.
- [2] R Core Team. *Writing R Extensions*. Version 2.15.1. 2012.
- [3] Faraway JJ. *Extending the Linear Model with R*. Chapman & Hall, 2006.
- [4] Agresti A. *Categorical Data Analysis*. Wiley, 2nd Ed. 2007.