Package 'glm.deploy'

May 8, 2026

Type Package

Title 'C' and 'Java' Source Code Generator for Fitted Glm Objects

Version 1.0.4

Date 2018-03-06

Author Oscar Castro-Lopez [cre, aut],
Ines Vega-Lopez [aut]

Maintainer Oscar Castro-Lopez <castroloj@gmail.com>

License GPL (>= 3) | file LICENSE

Description Provides two functions that generate source code implementing the predict function of fitted glm objects. In this version, code can be generated for either 'C' or 'Java'. The idea is to provide a tool for the easy and fast deployment of glm predictive models into production. The source code generated by this package implements two function/methods. One of such functions implements the equivalent to predict(type=``response"), while the second implements predict(type=``link"). Source code is written to disk as a .c or .java file in the specified path. In the case of c, an .h file is also generated.

URL https://github.com/oscarcastrolopez/glm.deploy

BugReports https://github.com/oscarcastrolopez/glm.deploy/issues

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.12), stats

LinkingTo Rcpp

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, testthat

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-03-09 09:59:35 UTC

Contents

`glm.deploy` *'C' and 'Java' Source Code Generator for Fitted Glm Objects*

Description

Provides two functions that generate source code implementing the predict function of fitted glm objects. In this version, code can be generated for either 'C' or 'Java'. The idea is to provide a tool for the easy and fast deployment of glm predictive models into production. The source code generated by this package implements two function/methods. One of such functions implements the equivalent to predict(type="response"), while the second implements predict(type="link"). Source code is written to disk as a .c or .java file in the specified path. In the case of c, an .h file is also generated.

Details

All numeric variables used as input to the glm object are treated as doubles, whereas factors variables are treated as strings.

The `glm.deploy` package is compatible with the following link functions in the fitted glm object:

- identity
- probit
- cloglog
- log
- sqrt
- inverse
- 1/mu²
- logit

Author(s)

- Oscar Castro-Lopez <castroloj@gmail.com>
- Ines Vega-Lopez <ifvega@uas.edu.mx>

See Also

Functions:

- glm2c
- glm2java

URL: https://github.com/oscarcastrolopez/glm.deploy

`glm2c` *C source code generator for rapid deployment of glm predictive models*

Description

The `glm2c()` function is used to generate source code in C language implementing a given glm predictive model. It implements the following two functions; the glm_xxx_response() and glm_xxx_link(), where xxx stands for the name of the target variable of the glm object.

After the invocation of the `glm2c()` function two files are generated:

- A .c file with the two scoring functions.
- An .h file with the prototypes of the two functions of the .c file.

Usage

```
glm2c(model, filename = NULL, path = NULL)
```

Arguments

`model`	A fitted object of class "glm".
`filename`	OPTIONAL The name of the output file(s), the default filenames are "glm_xxx.c" and "glm_xxx.h", where xxx is the target variable's name.
`path`	The directory path where files are going to be saved.

Note

All numeric variables used as input to the glm object are treated as doubles, whereas factor variables are treated as strings.

Author(s)

Oscar Castro-Lopez, Ines Vega-Lopez

See Also

glm2java

Examples

```
 # Example with the iris dataset with a Logical target and numeric variables,
 # using the binomial family and the logit link function
 data(iris)
 iristest = iris
 iristest$Virginica = ifelse(iristest$Species == 'virginica', TRUE,FALSE)
 iristest$Species = NULL

 # Load Package
 library(glm.deploy)
 # For repeatable results
 set.seed(123)
 # Generate the fitted glm object
 m = glm(Virginica ~ ., family = binomial(logit), data=iristest)
 # Call the glm2c() function with default filename
 glm2c(m,,tempdir())

 # Call the glm2c() function with custom filename
 glm2c(m,'my_glm_virginica', tempdir())

 # The glm2c() function generates the files: "glm_virginica.c" and
 # "glm_virginica.h"

## Not run:
---------------Contents of the "glm_virgninica.c" file---------------

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

double glm_virginica_link(double sepal_length,
                          double sepal_width,
                          double petal_length,
                          double petal_width){
  double new_sepal_length = -2.46522019518341 * sepal_length;
  double new_sepal_width = -6.68088701405762 * sepal_width;
  double new_petal_length = 9.4293851538836 * petal_length;
  double new_petal_width = 18.2861368877881 * petal_width;

  return -42.6378038127854+new_sepal_length+
                           new_sepal_width+
                           new_petal_length+
                           new_petal_width;
}
double glm_virginica_response(double sepal_length,
                              double sepal_width,
                              double petal_length,
```

```
                              double petal_width){
  return 1/(1+exp(-glm_virginica_link(sepal_length,
                                      sepal_width,
                                      petal_length,
                                      petal_width)));
}
----End of Contents of the "glm_virgninica.c" file------------------
--------------------------------------------------------------------

-----Contents of the "glm_virgninica.h" file------------------------
double glm_virginica_link(double sepal_length,
                          double sepal_width,
                          double petal_length,
                          double petal_width);
double glm_virginica_response(double sepal_length,
                              double sepal_width,
                              double petal_length,
                              double petal_width);
-----End of Contents of the "glm_virgninica.h" file-----------------
--------------------------------------------------------------------

Usage of the functions in another programs;
1) We need to add an include line #include "virginica_glm.h" to all
source files that use library definitions.
2) Link the .c file with the library object file.
    gcc -c glm_virginica.c
3) The following is an example file "test.c" to call the functions
and print the result:

-------------------"test.c"---------------
#include <stdio.h>
#include "glm_virgnica.h" //Added to call the scoring functions.

int main(int argc, char *argv[]){
  printf("%f\n",glm_virginica_link(5.7,2.5,5.0,2.0));
  printf("%f\n",glm_virginica_response(5.7,2.5,5.0,2.0));
  return 0;
}
---------------End of "test.c"---------------
---------------------------------------------

4) Compile the "test.c" file and link it to the glm_virginica shared
library, we also need to add the "-lm" option to link it to the
math.h library:
gcc test.c -o test glm_virginica.o -lm

5) Finally Run the test.o program in linux:
./test

## End(Not run)
```

glm2java *Java source code generator for rapid deployment of glm predictive models*

Description

The glm2java() function is used to generate source code in Java language implementing a given glm predictive model. It implements the following two methods; the glm_xxx_response() and glm_xxx_link(), where xxx stands for the name of the target variable of the glm object.

After invocation of the glm2java(), a .java file is generated containing the two predict methods which are declared as public static inside a java class called "glm_xxx_class".

Usage

```
glm2java(model, filename = NULL, path = NULL)
```

Arguments

model	A fitted object of class "glm".
filename	OPTIONAL The name of the output file, the default file name is "glm_xxx_class.java", where xxx is the target variable's name.
path	The directory path where files are going to be saved.

Note

All numeric variables used as input to the glm object are treated as doubles, whereas factors variables are treated as strings.

Author(s)

Oscar Castro-Lopez, Ines Vega-Lopez

See Also

glm2java

Examples

```
# Example with the iris dataset with a Logical target and numeric
# variables, using the binomial family and the logit link function
data(iris)
iristest = iris
iristest$Virginica = ifelse(iristest$Species == 'virginica', TRUE,FALSE)
iristest$Species = NULL

# Load Package
library(glm.deploy)
```

```
 # For repeatable results
 set.seed(123)
 # Generate the fitted glm object
 m = glm(Virginica ~ ., family = binomial(logit), data=iristest)
 # Call the glm2java() function with default filename
 glm2java(m,, tempdir())
 # Call the glm2java() function with custom filename
 glm2java(m,'my_glm_virginica', tempdir())

 # The glm2java() function generates the file "glm_virginica_class.java".

## Not run:
----------Contents of the "glm_virgninica_class.java" file-------
  package test;
  public class glm_virginica_class{

  public static double glm_virginica_link(double sepal_length,
                                          double sepal_width,
                                          double petal_length,
                                          double petal_width){
      double new_sepal_length = -2.46522019518341 * sepal_length;
      double new_sepal_width = -6.68088701405762 * sepal_width;
      double new_petal_length = 9.4293851538836 * petal_length;
      double new_petal_width = 18.2861368877881 * petal_width;

      return -42.6378038127854+new_sepal_length+
                               new_sepal_width+
                               new_petal_length+
                               new_petal_width;
    }
    public static double glm_virginica_response(double sepal_length,
                                                double sepal_width,
                                                double petal_length,
                                                double petal_width){
      return 1/(1+Math.exp(-glm_virginica_link(sepal_length,
                                              sepal_width,
                                              petal_length,
                                              petal_width)));
    }

  }
---------------End of "glm_virgninica_class.java"---------------
----------------------------------------------------------------
To use these methods in another class just add
the "import glm_virginica_class.*;"

## End(Not run)
```

Index