Package ‘StepBeta’

October 12, 2022

Type  Package
Title  Stepwise Procedure for Beta, Beta-Binomial and Negative Binomial
       Regression Models
Version  2.1.0
Author  Sergio Garofalo
Maintainer  Sergio Garofalo <sergio.garofalo96@gmail.com>
Description  Starting from a Regression Model, it provides a stepwise
              procedure to select the linear predictor.
License  GPL-3
Encoding  UTF-8
RoxygenNote  7.2.1
Imports  glue, stats, betareg, combinat, aod, MASS
NeedsCompilation  no
Repository  CRAN
Date/Publication  2022-08-10 15:30:20 UTC

R topics documented:

  check_formula_terms .............................................. 2
  Combination_Terms ................................................. 2
  dispersion_formula_terms .......................................... 3
  keep_formula_terms ............................................... 3
  remove_formula_interactions .................................... 4
  StepBeta ............................................................. 4
  Step_glimML .......................................................... 6

Index  9
**check_formula_terms**  
*StepBeta and StepBetaBinomial internal object*

**Description**
StepBeta and StepBetaBinomial internal object

**Usage**

```r
check_formula_terms(model)
```

**Arguments**
- `model`  
  Beta regression model

**Value**
It returns the complete formula in a standard form

---

**Combination_Terms**  
*StepBetaBinomial internal object*

**Description**
StepBetaBinomial internal object

**Usage**

```r
Combination_Terms(Terms, interaction = F)
```

**Arguments**
- `Terms`  
  Variables from the starting model
- `interaction`  
  Parameter to define which part of linear predictor to operate

**Value**
The function create alle possible combination of the linear predictor
**dispersion_formula_terms**

*StepBeta internal object*

**Description**

StepBeta internal object

**Usage**

```
dispersion_formula_terms(object)
```

**Arguments**

- **object**
  - full model

**Value**

The function updates the formula for the dispersion component of the model

---

**keep_formula_terms**

*StepBeta internal object*

**Description**

StepBeta internal object

**Usage**

```
keep_formula_terms(the_formula, var_name)
```

**Arguments**

- **the_formula**
  - Formula of Beta Regression model
- **var_name**
  - Names of the variables to keep

**Value**

The function updates the formula, it keeps the variables defined by the user
remove_formula_interactions

StepBeta internal object

Description
StepBeta internal object

Usage
remove_formula_interactions(the_formula)

Arguments
the_formula    Formula of Beta Regression model

Value
The function returns a reduced form of the formula. It excludes the interactive effects.

StepBeta

Stepwise model selection for Beta Regression

Description
This function performs a stepwise algorithm to define the best linear predictor according to an user defined criterion (default is the Akaike Information Criterion aka AIC). It works for objects of class "betareg" from betareg function. If the object is different from "betareg" class, the function performs the classical step function in "stats" package.

Usage
StepBeta(object, k = 2, dispersion = T)

Arguments
object    Object of class "betareg". If the class is different the function apply the step function in "stats" package
k        The penalty parameter used for the criterion, e.g. default is k = 2 which identify the classical AIC. BIC can be obtained as k = log(n)
dispersion        Provide the stepwise procedure also for dispersion parameter. Default is TRUE
StepBeta is different from step (stats) and stepAIC (MASS) functions; for an object of class "betareg" is impossible to use an algorithm which uses the function extractAIC Starting from a full model it provides a backward procedure where the scope model is the reduced one.

First, StepBeta operates with all the principal effects included in the model; starting from the full model, the algorithm computes all the possible models, it calculates the measure (default is AIC) and it defines as a good predictor the model with lower AIC.

Then, based on the previous results, StepBeta operates adding all the possible interactive effects. As in the first passage, the model chosen by the algorithm is the one whose AIC is the lowest.

During the procedure, StepBeta considers all the possible models which betareg can fit. There are many cases where betareg function falls into error, in these cases the algorithm does not consider the linear predictor which causes the error and it goes forward.

The algorithm returns an object of class "betareg"

Sergio Garofalo


library(betareg)
data <- iris
data$Sepal.Length <- data$Sepal.Length/(max(data$Sepal.Length) + 0.01)

fullModel <- betareg(Sepal.Length ~ Sepal.Width * Petal.Length * Petal.Width * Species, data = data)
reducedModel <- StepBeta(fullModel)
summary(reducedModel)
### Mean and precision parameters

```r
reducedModel <- StepBeta(fullModel, dispersion = TRUE)
summary(reducedModel)
```

---

**Description**

This function performs a stepwise algorithm to define the best linear predictor according to an user defined criterion (default is the Akaike Information Criterion aka AIC, but it is also possible to perform the corrected version AICc). It works only for object from betabin function (class "glimML" from "aod" package). If the object is different from "glimML" class, the function performs the classical step function in "stats" package.

**Usage**

```
Step_glimML(object, k = 2, overdispersion = T, correctAIC = T)
```

**Arguments**

- `object`: Object of class "glimML". If the class is different the function apply step function in "stats" package
- `k`: The penalty parameter used for the criterion, e.g. default is `k = 2` which identify the classical AIC. BIC can be obtained as `k = log(n)`
- `overdispersion`: Provide the stepwise procedure also for the overdispersion component of the model (defined as random) Default is TRUE
- `correctAIC`: Use AICc instead of AIC. Default TRUE is for AICc

**Details**

`Step_glimML` is different from `step` (stats) and `stepAIC` (MASS) functions; for an object of class betabin is impossible to use an algorithm which uses the function `extractAIC`. Starting from a full model it provides a backaward procedure where the scope model is the reduced one.

First, `Step_glimML` operates with all the principal effects included in the model; starting from the full model, the algorithm computes all the possible models, it calculates the measure (default is AIC) and it defines as a good predictor the model with lower AIC.

Then, based on the previous results, `Step_glimML` operates adding all the possible interactive effects. As in the first passage, the model choosen by the algorithm is the one whose AIC is the lowest.
During the procedure, Step_glimML considers all the possible models which `betabin` can fit. There are many cases where `betabin` function falls into error, in these cases the algorithm does not consider the linear predictor which causes the error and it goes forward.

**Value**

The algorithm returns an object of class "`glimML`"

**Author(s)**

Sergio Garofalo

**References**


**Examples**

```r
# Starting from a "betabinom" model
# Wadsworth & Brooks/Cole. (has iris3 as iris.)

# Prepare the data
library(aod)
data(iris)

############ Beta Binomial model
# Not run:
n <- round(runif(dim(iris)[1],1,50))
y <- round(runif(length(n), 1,n))
data <- cbind(iris,y,n)
fullModel <- betabin(cbind(y, n - y) ~ Sepal.Width * Petal.Length + Petal.Width, ~ Species, data = data)
reducedModel <- Step_glimML(fullModel)
summary(reducedModel)

############ Negative Binomial model
# Not run:
data$Sepal.Length <- round(Sepal.length + runif(dim(data)[1],0,1) * 100)
fullModel <- negbin(Sepal.Length ~ Sepal.Width * Petal.Length + Petal.Width, ~ Species, data = data)
reducedModel <- Step_glimML(fullModel)
```

summary(reducedModel)

## End(Not run)
Index

check_formula_terms, 2
Combination_Terms, 2

dispersion_formula_terms, 3
keep_formula_terms, 3
remove_formula_interactions, 4

Step_glimML, 6
StepBeta, 4