# Package: cowbell (via r-universe)

August 30, 2024

**Type** Package

**Title** Performs Segmented Linear Regression on Two Independent Variables

**Version** 0.1.0

**Description** Implements a specific form of segmented linear regression with two independent variables. The visualization of that function looks like a quarter segment of a cowbell giving the package its name. The package has been specifically constructed for the case where minimum and maximum value of the dependent and two independent variables are known a prior, which is usually the case when those values are derived from Likert scales.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** ggplot2, rgl, grDevices, misc3d

**RoxygenNote** 6.0.1

**Depends** R (>= 3.3.0)

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Christoph Luerig [aut, cre]

**Maintainer** Christoph Luerig <luerig@hochschule-trier.de>

**Repository** CRAN

**Date/Publication** 2017-05-05 21:01:45 UTC

## Contents

---

`allFun` *allFun: Data set for Fluency, Absorption, Fit and Fun.*

---

## Description

Data set of two games containing measurements of Fluency, Absorption, Fit (= Perceived difficulty) and Fun. Fluency and Absorption are measured on a scale from 1..7, Fun and Fit are measured 1..9.

---

`cowbell` *Computes a response surface as segmented linear regression that resembles a cowbell.*

---

## Description

The application case for which this package has been constructed for is the response analysis of one dependend variable that depends on two independend variable. Additionally a prior information is given on the minimal and maximal values of those three variables. This is the case if those variables originate from questions using Likert scales. This information is indicated upfront with the function `generateCowbellConcept`. The response surface resembles a quarter segment of a cowbell. It starts with a certain value at the dependend variable if any of the independent variables are at their minimal value. If both independent variables have reached their breakpoint value a new value with a plateau will be returned. Going to the plateau a rising linear ridge is used. Where to both sides of the ridge there is only a linear dependency on one of the independent variables. In general this model may be worth trying if one assumes that there is a point of saturation for both independent variables and that up this point essentially the smaller one of both values is determining the result on the dependend variable. This is computed in a second step with the help of the function `generateCowbell`.

## fitted.cowbell *Implementation of the* fitted *generic.*

### Description

Implementation of the fitted generic.

### Usage

```
## S3 method for class 'cowbell'
fitted(object, ...)
```

### Arguments

| | |
|---|---|
| object | The data obtained by function generateCowbell. |
| ... | Just for compatibility purposes. |

### Value

List with predicted values

### Examples

```
# Run a simplified anaylsis with 10 iterations only (to save time.)
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
data(allFun)
test<-generateCowbell(concept, allFun, 10)
fitted(test)
```

## generateCowbell *Performs the segmented linear regression analysis generating the cowbell function.*

### Description

This function takes the cowbell definition that was created with generateCowbellConcept and performs a regression analysis. Additionally it also moves the breaking point to the maximal values of the independent variables to later on test for the significance of the breaking point. This function needs relatively long to compute as it uses a gradient based optimizer for optimizing a non - linear model.

### Usage

```
generateCowbell(concept, table, iterations = 1000, learningRate = 0.01)
```

### Arguments

| | |
|---|---|
| concept | The previously in function generateCowbellConcept specified concept. |
| table | The table that at least contains the data for the dependent and the two independent variables specified in concept. |
| iterations | The number of iteration that should be done with the gradient optimizer. |
| learningRate | The step size that should be applied in the gradient optimizer. |

### Value

A list with the data, the model with and without breakpoint and the F-Statistics.

### See Also

generateCowbellConcept

### Examples

```
# Run a simplified anaylsis with 10 iterations only (to save time.)
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
data(allFun)
test<-generateCowbell(concept, allFun, 10)
```

---

## generateCowbellConcept

*Expresses the fitting formula and the value range of the variables.*

---

### Description

Generates a concept which is basically the formula used for the regression analysis and minimal and maximal values of the dependent and two independent variables. As this package creates a very specific regression model the formula is always of the form Dest ~ SrcA * SrcB. If one of the minimal and maximal values are omitted, the minimum or maximum value of the data set will be used later on.

### Usage

```
generateCowbellConcept(formula, minDest = NA, maxDest = NA, minSrcA = NA,
  maxSrcA = NA, minSrcB = NA, maxSrcB = NA)
```

### Arguments

| | |
|---|---|
| formula | The formula essentially specifying the names of the dependend variable (here Dest) and the two independent variables (here SrcA, SrcB). Example: Dest ~ SrcA * SrcB. |
| minDest | The a prior known minimal value the dependend variable (Dest) can have. |
| maxDest | The a prior known maximal value the dependend variable (Dest) can have. |

| | |
|---|---|
| minSrcA | The a prior known minimal value the first dependend variable (SrcA) can have. |
| maxSrcA | The a prior known maximal value the first dependend variable (SrcA) can have. |
| minSrcB | The a prior known minimal value the second dependend variable (SrcB) can have. |
| maxSrcB | The a prior known maximal value the second dependend variable (SrcB) can have. |

## Value

List of the aggregate information. This is used in the regression analysis.

## Examples

```
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
```

---

# plot.cowbell *Plots the obtained cowbell function.*

---

## Description

Generates a three dimension plot of cowbell function. Additionally the data points of the original data set are added in the visualization. The function with and without breaking point can be visualized. Implementation of the plot generic.

## Usage

```
## S3 method for class 'cowbell'
plot(x, breakPointUsed = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | The data obtained by function generateCowbell. |
| breakPointUsed | Defaults to TRUE and indicates if we want to use the version with breakpoint (or not). |
| ... | Just for compatibility purposes. |

## Examples

```
# Run a simplified anaylsis with 10 iterations only (to save time.)
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
data(allFun)
test<-generateCowbell(concept, allFun, 10)
plot(test)
```

predict.cowbell *Performs a prediction on the cowbell model that has been generated.*

## Description

Implementation of the predict generic. The provided data has to have the exact column names that were used when the cowbell analysis was done. If no data is provided the original data is used.

## Usage

```
## S3 method for class 'cowbell'
predict(object, newdata, ...)
```

## Arguments

| | |
|---|---|
| object | The data obtained by function generateCowbell. |
| newdata | The data set to perform the prediction on. If omitted the original data is used. |
| ... | Just for compatibility purposes. |

## Value

The vector with the predicted data.

## Examples

```
# Run a simplified anaylsis with 10 iterations only (to save time.)
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
data(allFun)
test<-generateCowbell(concept, allFun, 10)
predict(test)
```

print.cowbell *Summarizes the cowbell regression analysis*

## Description

Prints the used formula, the R squared and the F statistics in comparison with a constant function (average of values).

## Usage

```
## S3 method for class 'cowbell'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object generated with function generateCowbell |
| ... | Only for compatibility purposes. |

### Details

Implementation of the print generic.

### Examples

```
# Run a simplified anaylsis with 10 iterations only (to save time.)
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
data(allFun)
test<-generateCowbell(concept, allFun, 10)
test
```

---

## print.cowbellConcept *Summarizes the cowbell concept with the formula and value ranges.*

---

### Description

Implementation of the print generic.

### Usage

```
## S3 method for class 'cowbellConcept'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object generated with function generateCowbellConcept |
| ... | Unused for compatibility only. |

### See Also

generateCowbellConcept

### Examples

```
# Generate a concept and display it.
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
concept
```

`print.summary.cowbell` *Prints the summary obtained by* `summary.cowbell`.

## Description

The output states the concept consisting of the formula and the value range, that was given as a prior information. Then follows the characteristics of the cowbell function. This is the minimal and maximal value of the dependent variable. The maximum is reached at the plateau part and the minimal at the outer ring of the cowbell. It gets reached if any of the two independent variables have reached their minimum. The R function with which the cowbell gets computed is then appended. It follows the R squared and F-statistics in comparison with a constant function.

## Usage

```
## S3 method for class 'summary.cowbell'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | The object to print generated by `summary.cowbell` |
| `...` | Just for compatibility purposes. |

## Details

The following analysis is done to check the significance of the breakpoint. The breakpoint gets eliminated by removing the plateau. The linear rising ridge of the cowbell is rised up to the specified maximum of the independent variables. Therefore only the minimal and maximal value of the dependent variable is left of the definition. What follows is the string that characterizes the R function and the R Squared of the used model. The following F-Statistics compares the full model with the breakpoint against this reduced model without to estimate the significance of the breakpoint.

Implementation of the `print` generic.

## Examples

```
# Run a simplified anaylsis with 10 iterations only (to save time.)
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
data(allFun)
test<-generateCowbell(concept, allFun, 10)
summary(test)
```

## residuals.cowbell *Implementation of the* residuals *generic.*

### Description

Implementation of the residuals generic.

### Usage

```
## S3 method for class 'cowbell'
residuals(object, ...)
```

### Arguments

| | |
|---|---|
| object | The data obtained by function generateCowbell. |
| ... | Just for compatibility purposes. |

### Value

Vector with the residuals to the data.

### Examples

```
# Run a simplified anaylsis with 10 iterations only (to save time.)
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
data(allFun)
test<-generateCowbell(concept, allFun, 10)
residuals(test)
```

## summary.cowbell *Generates the core information of the cowbell analysis.*

### Description

Generates a list that contains relevant information of the cowbell analysis. concept contains the formula and the minimal and maximal values. baseDefinition contains the information for the cowbell. This is the minimal and the maximal value the cowbell can reach and the position of the breaking point. baseDefinitionReduced only contains the minimal and the maximal value, the breakpoint is not included in that model. functionString is a string version of the R function that generates the cowbell. functionStringReduced is the string version without the breaking point. fstatistic contains the fstatistic information of the cowbell model in contrast to the constant function. fstatisticBreakpoint describes the F-statistics of the full cowbell model against the version without break point.

## Usage

```
## S3 method for class 'cowbell'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| object | The resulting object of generateCowbell. |
| ... | Just for compatibility purposes. |

## Details

Implementation of the summary generic.

## Value

List with the mentioned values.

## Examples

```
# Run a simplified anaylsis with 10 iterations only (to save time.)
concept<-generateCowbellConcept(Fun ~ Fluency * Absorption, 1, 9, 1, 7, 1, 7)
data(allFun)
test<-generateCowbell(concept, allFun, 10)
summary(test)$functionString
```

---

# testA *testA: Artificial data set for testing.*

---

## Description

This is an artificial data with a perfect cowbell structure that is used for testing.

# Index