Package 'colf'

October 9, 2017

Type Package

Title Constrained Optimization on Linear Function

Version 0.1.3

URL https://github.com/LyzandeR/colf

BugReports https://github.com/LyzandeR/colf/issues

Depends R (>= 3.2.0), nlsr, stats, utils

Description Performs least squares constrained optimization on a linear objective function. It contains a number of algorithms to choose from and offers a formula syntax similar to lm().

License MIT + file LICENSE

LazyData TRUE

RoxygenNote 5.0.1

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Theo Boutaris [aut, cre, cph]

Maintainer Theo Boutaris <theo.boutaris@lyzander.com>

Repository CRAN

Date/Publication 2017-10-09 20:24:44 UTC

R topics documented:

`coef.colf_nlxb` *Coefficients for colf_nlxb*

Description

Coefficients for colf_nlxb

Usage

```
## S3 method for class 'colf_nlxb'
coef(object, ...)
```

Arguments

`object`	A colf_nlxb object i.e. the result of running `colf_nlxb`
`...`	Currently not used

Value

A vector with the coefficients

Examples

```
mymod <- colf_nlxb(mpg ~ hp + cyl, mtcars)

#coefficients
coef(mymod)
```

`colf_nls` *Non linear Least Squares Optimization on a Linear Objective Function*

Description

Non linear least squares optimization using the port algorithm on a linear objective function.

Usage

```
colf_nls(formula, data, start = NULL, trace = FALSE, control = NULL,
  na.action = c("na.omit", "na.fail", "na.exclude"), lower = -Inf,
  upper = Inf, ...)
```

Arguments

formula	The formula. This has the same syntax and supports the same features as the formula in lm. See examples.
data	A data frame containing the data of the variables in the formula.
start	An atomic vector of same length as the number of parameters. If not provided a cheap guess will be made. If categorical variables are included these need to be takent into consideration as number of categories minus one. See examples and details.
trace	Logical. Defaults to FALSE. Set to TRUE if you want the intermediate progress to be reported
control	an optional list of control settings. See nls.control for the names of the settable control values and their effect.
na.action	A function which indicates what should happen if NAs are present in the data set. Defaults to options('na.action'). na.fail, or na.exclude can be used.
lower	Lower bounds of the parameters (atomic vector). If a single number, this will be applied to all parameters. Defaults to -Inf (unconstrained).
upper	Upper bounds of the parameters (atomic vector). If a single number, this will be applied to all parameters. Defaults to Inf (unconstrained).
...	Other arguments passed on to optimiser

Details

colf_nls uses nls, in an attempt to find the minimum of the residual sum of squares. The algorithm is applied on a linear objective function.

The function provides an easy way to apply the optimizer on a linear objective function in a similar way to lm.

start, lower and upper, if provided, can be either an atomic vector which has the same length as the number of parameters or a single number which will be replicated to match the length of the parameters. If categorical variables exist in the function these will be dummified. Out of one categorical variable, n - 1 will be created where n is the total number of categories in the variable. This needs to be taken into account when providing an atomic vector for start, lower or upper. Also, as with lm an intercept will be added which also needs to be taken into account.

Value

Same as nls

See Also

nls, nls.control

Examples

```
#no constraints
colf_nls(mpg ~ cyl + disp, mtcars)
```

```
#no intercept
colf_nls(mpg ~ 0 + cyl + disp, mtcars)

#including categorical variables. These will be dummified.
colf_nls(Sepal.Length ~ Sepal.Width + Species, iris)

#lower boundary will be replicated for all parameters
colf_nls(Sepal.Length ~ Sepal.Width + Species, iris, lower = 0.5)

#species is categorical and contains 3 categories, thus we need to specify 4 lower bounds:
#the first one for the intercept.
#the second one for Sepal.Width
#the two next for the dummy variables constructed from Species.
colf_nls(Sepal.Length ~ Sepal.Width + Species, iris, lower = rep(0.5, 4))
```

`colf_nlxb` *Nash Variant of the Marquardt algorithm on a linear objective function*

Description

Non linear least squares solution via qr linear solver on a linear objective function.

Usage

```
colf_nlxb(formula, data, start = NULL, trace = FALSE, lower = -Inf,
  upper = Inf, na.action = c("na.omit", "na.fail", "na.exclude"),
  masked = NULL, control = NULL, ...)
```

Arguments

`formula`	The formula. This has the same syntax and supports the same features as the formula in `lm`. See examples.
`data`	A data frame containing the data of the variables in the formula.
`start`	An atomic vector of same length as the number of parameters. If not provided a cheap guess will be made. If categorical variables are included these need to be takent into consideration as number of categories minus one. See examples and details.
`trace`	Logical. Defaults to FALSE. Set to TRUE if you want the intermediate progress to be reported
`lower`	Lower bounds of the parameters (atomic vector). If a single number, this will be applied to all parameters. Defaults to -Inf (unconstrained).
`upper`	Upper bounds of the parameters (atomic vector). If a single number, this will be applied to all parameters. Defaults to Inf (unconstrained).
`na.action`	A function which indicates what should happen if NAs are present in the data set. Defaults to options('na.action'). na.fail, or na.exclude can be used.

`masked` Character vector of parameter names. These parameters will not be altered by the algorithm.

`control` A list of controls for the algorithm. These are:

- watch - Monitor progress. Logical, defaults to FALSE
- phi - Adds phi*identity to Jacobian inner product. Defaults to 1.
- lamda - Initial Marquardt adjustment. Defaults to 0.0001.
- offset - Shift to test floating point equality. Defaults to 100.
- laminc - Factor to use to increase lamda. Defaults to 10.
- lamdec - Factor to use to decrease lamda (lamdec / laminc). Defaults to 4.
- femax - Maximum evaluations of sum of squares function. Defaults to 10000.
- jemax - Maximum evaluations of the Jacobian. Defaults to 5000.
- rofftest - Use a termination of the relative offset orthogonality type.
- smallsstest - Exit the function if the sum of squares falls below (100 * .Machine$double.eps)^4 times the initial sumsquares. Defaults to TRUE.

`...` Other arguments passed on to optimiser

Details

`colf_nlxb` uses Nash's (Nash, 1979) variant of the Marquardt algorithm, in an attempt to find the minimum of the residual sum of squares. The algorithm is applied on a linear objective function.

The function provides an easy way to apply the optimizer on a linear objective function in a similar way to `lm`.

start, lower and upper, if provided, can be either an atomic vector which has the same length as the number of parameters or a single number which will be replicated to match the length of the parameters. If categorical variables exist in the function these will be dummified. Out of one categorical variable, n - 1 will be created where n is the total number of categories in the variable. This needs to be taken into account when providing an atomic vector for start, lower or upper. Also, as with `lm` an intercept will be added which also needs to be taken into account.

Value

Same as nlxb

See Also

nlxb

Examples

```
#no constraints
colf_nlxb(mpg ~ cyl + disp, mtcars)

#no intercept
colf_nlxb(mpg ~ 0 + cyl + disp, mtcars)

#including categorical variables. These will be dummified.
```

```
colf_nlxb(Sepal.Length ~ Sepal.Width + Species, iris)

#lower boundary will be replicated for all parameters
colf_nlxb(Sepal.Length ~ Sepal.Width + Species, iris, lower = 0.5)

#species is categorical and contains 3 categories, thus we need to specify 4 lower bounds:
#the first one for the intercept.
#the second one for Sepal.Width
#the two next for the dummy variables constructed from Species.
colf_nlxb(Sepal.Length ~ Sepal.Width + Species, iris, lower = rep(0.5, 4))
```

`construct_formula` *Construct an nls-compatible formula from an lm style formula*

Description

Construct an nls-compatible formula from an lm style formula

Usage

```
construct_formula(formula, data)
```

Arguments

`formula`	The formula. This has the same syntax and supports the same features as the formula in `lm`. See examples.
`data`	A data frame containing the data of the variables in the formula.

Details

`construct_formula` creates the parameters needed for the formula to be compatible with nls style functions. It also creates and returns the modelling set.

`construct_formula` will make syntactically valid names (if applicable) otherwise the optimizers will fail. To make these names `make.names` is used. Check examples.

Value

A list of three elements:

- model_formula - An nls compatible formula
- model_data - The modelling set created (inlcuding dummy variables, if any)
- x_param_names - The names of the parameters

See Also

nls, make.names

Examples

```
#simple syntax
construct_formula(mpg ~ hp + cyl, mtcars)

#example of make.names to create syntactically valid names
make.names('(foo/^@bar)')

#function will create syntactically valid names (if applicable)
#otherwise the optimizers will fail
construct_formula(mpg ~ I(hp + cyl), mtcars)
construct_formula(mpg ~ (hp + cyl + disp)^3, mtcars)
```

`fitted.colf_nlxb` *Fitted values for colf_nlxb*

Description

Fitted values for colf_nlxb

Usage

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

Arguments

`object`	A colf_nlxb object i.e. the result of running `colf_nlxb`
`...`	Currently not used

Value

A vector with the fitted values

Examples

```
mymod <- colf_nlxb(mpg ~ hp + cyl, mtcars)

#fitted values
fitted(mymod)
```

`predict.colf_nls` *Predict method for colf_nls*

Description

Predict method for colf_nls

Usage

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

Arguments

`object`	A colf_nls object
`newdata`	A new data.frame which contains the same column names and classes as the original data.frame
`...`	Currently not used

Details

`predict.colf_nls` will use the fit model to predict on a new data set.

When using predict.colf_nls make sure the column names and classes of the new data set are the same as the data the model was trained on.

Value

A vector with the predictions

Examples

```
mymod <- colf_nls(mpg ~ hp + cyl, mtcars)

#prediction
predict(mymod, mtcars)
```

`predict.colf_nlxb` *Predict method for colf_nlxb*

Description

Predict method for colf_nlxb

Usage

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

Arguments

`object`	A colf_nls object
`newdata`	A new data.frame which contains the same column names and classes as the original data.frame
`...`	Currently not used

Details

`predict.colf_nlxb` will use the fit model to predict on a new data set.

When using predict.colf_nlxb make sure the column names and classes of the new data set are the same as the data the model was trained on.

Value

A vector with the predictions

Examples

```
mymod <- colf_nlxb(mpg ~ hp + cyl, mtcars)

#prediction
predict(mymod, mtcars)
```

print.colf_nlxb *colf_nlxb Print method*

Description

colf_nlxb Print method

Usage

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

Arguments

x	A colf_nlxb object i.e. the result of running colf_nlxb
...	Currently not used

Value

Printing the colf_nlxb object

Examples

```
mymod <- colf_nlxb(mpg ~ hp + cyl, mtcars)

#print
print(mymod)
```

residuals.colf_nlxb *Residuals for colf_nlxb*

Description

Residuals for colf_nlxb

Usage

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

Arguments

object	A colf_nlxb object i.e. the result of running colf_nlxb
...	Currently not used

Value

A vector with the residuals

Examples

```
mymod <- colf_nlxb(mpg ~ hp + cyl, mtcars)

#residuals
residuals(mymod)
resid(mymod)
```

`summary.colf_nlxb` *colf_nlxb Summary*

Description

colf_nlxb Summary

Usage

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

Arguments

`object`	A colf_nlxb object i.e. the result of running `colf_nlxb`
`...`	Currently not used

Value

The summary of the model

Examples

```
mymod <- colf_nlxb(mpg ~ hp + cyl, mtcars)

#summary
summary(mymod)
```

Index