Package 'glmgen'

September 5, 2014

Type Package

Title Fast generalized lasso solver

Version 0.0.1

Date 2014-09-03

Author Taylor B. Arnold, Ryan Tibshirani

Maintainer Taylor Arnold <taylor.arnold@aya.yale.edu>

Description An efficent implementation of several numerical algorithms
for solving the generalized lasso. Specialized implementations are provided
to deal with special cases, such as trendfiltering and the (graph) fused lasso.

License GPL-2 | GPL-3

Depends methods

R topics documented:

glmgen-package *Fast generalized lasso solver*

Description

An efficent implementation of several numerical algorithms for solving the generalized lasso. Specialized implementations are provided to deal with special cases, such as trendfiltering and the (graph) fused lasso.

Details

Package:	glmgen
Type:	Package
Version:	0.0.1
Date:	2014-09-03
License:	GPL-2 | GPL-3
Depends:	methods

Author(s)

Taylor B. Arnold, Ryan Tibshirani

Maintainer: Taylor Arnold <taylor.arnold@aya.yale.edu>

coef-methods *Coefficients – S4 Methods*

Description

The "glmgen" package provides methods for calculating the coefficients of a generalized lasso fit via the generic function coef.

Methods

```
signature(object = "ANY")
```

```
signature(object = "glmgen")
```

glmgen-class *Class* "glmgen"

Objects from the Class

Objects can be created by calls of the form `new("glmgen", ...)`.

Slots

y: Object of class "numeric" ~~

lambda: Object of class "numeric" ~~

beta: Object of class "matrix" ~~

family: Object of class "character" ~~

method: Object of class "character" ~~

n: Object of class "integer" ~~

p: Object of class "integer" ~~

d: Object of class "integer" ~~

obj: Object of class "numeric" ~~

call: Object of class "call" ~~

Methods

coef `signature(object = "glmgen")`: ...

print `signature(x = "glmgen")`: ...

show `signature(object = "glmgen")`: ...

Examples

```
showClass("glmgen")
```

predict-methods ~~ *Methods for Function* predict ~~

Description

~~ Methods for function predict ~~

Methods

```
signature(object = "ANY")
signature(object = "trendfilter")
```

print-methods ~~ *Methods for Function* print ~~

Description

~~ Methods for function print ~~

Methods

signature(x = "ANY")
signature(x = "glmgen")
signature(x = "summary.glmgen")

show-methods ~~ *Methods for Function* show ~~

Description

~~ Methods for function show ~~

Methods

signature(object = "ANY")
signature(object = "classGeneratorFunction")
signature(object = "classRepresentation")
signature(object = "envRefClass")
signature(object = "genericFunction")
signature(object = "genericFunctionWithTrace")
signature(object = "glmgen")
signature(object = "MethodDefinition")
signature(object = "MethodDefinitionWithTrace")
signature(object = "MethodSelectionReport")
signature(object = "MethodWithNext")
signature(object = "MethodWithNextWithTrace")
signature(object = "namedList")
signature(object = "ObjectsWithPackage")
signature(object = "oldClass")
signature(object = "refClassRepresentation")
signature(object = "refMethodDef")
signature(object = "refObjectGenerator")
signature(object = "signature")
signature(object = "sourceEnvironment")
signature(object = "summary.glmgen")
signature(object = "traceable")

summary-methods ~~ *Methods for Function* summary ~~

Description

~~ Methods for function summary ~~

Methods

signature(object = "ANY")

signature(object = "trendfilter")

summary.glmgen-class *Class* "summary.glmgen"

Objects from the Class

Objects can be created by calls of the form new("summary.glmgen", ...).

Slots

summary: Object of class "matrix" ~~

Methods

print signature(x = "summary.glmgen"): ...

show signature(object = "summary.glmgen"): ...

Examples

```
showClass("summary.glmgen")
```

trendfilter *a*

Usage

```
trendfilter(y, x, k = 0L, family = c("gaussian", "logistic", "poisson"), lambda, nlambda = 100L, la
```

Arguments

y

x

k

family

lambda

nlambda
lambda.min.ratio

method

maxiter

objective

control

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (y, x, k, family = c("gaussian", "logistic", "poisson"),
    lambda, nlambda = 100L, lambda.min.ratio = 1e-05, method = c("admm",
        "prime_dual"), maxiter = 1e+06, objective = FALSE, control = list())
{
    cl = match.call()
    n = length(y)
    nlam = as.integer(nlambda)
    family = match.arg(family)
    method = match.arg(method)
    family_cd = match(family, c("gaussian", "logistic", "poisson")) -
        1L
    method_cd = match(method, c("admm", "prime_dual")) - 1L
    if (k < 0 || k != floor(k))
        stop("k must be a nonnegative integer.")
    if (n < k + 2)
        stop("y must have length >= k+2 for kth order trend filtering.")
    if (maxiter <= 1L)
        stop("maxiter must be greater than 1")
    if (missing(lambda)) {
        if (nlam <= 0L)
            stop("nlambda must be a positive number.")
        if (lambda.min.ratio < 0 | lambda.min.ratio > 1)
            stop("lamba.min.ratio must be between 0 and 1.")
    }
    else {
        if (length(lambda) == 0L)
            stop("Must specify at least one lambda value.")
        if (min(lambda) < 0L)
            stop("All specified lambda values must be nonnegative.")
        nlambda = length(lambda)
    }
```

```
    if (!is.list(control) | (is.null(names(control)) & length(control) !=
        0L))
        stop("control must be a named list")
    control = lapply(control, function(v) ifelse(is.numeric(v),
        as.double(v[[1]]), stop("Elements of control must be numeric.")))
    if (is.null(x))
        x = 1L:length(y)
    if (missing(lambda)) {
        lambda = rep(0, nlambda)
        lambda_flag = FALSE
    }
    else lambda_flag = TRUE
    z = .Call("tf_R", sY = as.double(y), sX = as.double(x), sN = length(y),
        sK = as.integer(k), sFamily = as.integer(family_cd),
        sMethod = as.integer(method_cd), sMaxIter = as.integer(maxiter),
        sLamFlag = as.integer(lambda_flag), sObjFlag = as.integer(objective),
        sLambda = as.double(lambda), sNlambda = as.integer(nlambda),
        sLambdaMinRatio = as.double(lambda.min.ratio), sControl = control,
        package = "glmgen")
    if (is.null(z))
        stop("Unspecified error in C code.")
    if (!is.null(z$obj))
        z$obj = z$obj[1:z$numiter]
    else z$obj = NA_real_
    colnames(z$beta) = as.character(round(z$lambda, 3))
    out = new("trendfilter", y = y, x = x, k = as.integer(k),
        lambda = z$lambda, beta = z$beta, family = family, method = method,
        n = length(y), p = length(y), d = length(y) - as.integer(k) -
            1L, obj = z$obj, call = cl)
    out
}
```

`trendfilter-class` *Class* `"trendfilter"`

Objects from the Class

Objects can be created by calls of the form `new("trendfilter", ...)`.

Slots

`y`: Object of class `"numeric"` ~~

`x`: Object of class `"numeric"` ~~

`k`: Object of class `"integer"` ~~

`lambda`: Object of class `"numeric"` ~~

`beta`: Object of class `"matrix"` ~~

`family`: Object of class `"character"` ~~

`method`: Object of class `"character"` ~~

`n`: Object of class `"integer"` ~~

`p`: Object of class `"integer"` ~~

`d`: Object of class `"integer"` ~~

`obj`: Object of class `"numeric"` ~~

`call`: Object of class `"call"` ~~

Extends

Class "glmgen", directly.

Methods

predict `signature(object = "trendfilter")`: ...

summary `signature(object = "trendfilter")`: ...

Examples

```
showClass("trendfilter")
```

Index