Package ‘robustgam’

February 20, 2015

Type Package
Title Robust Estimation for Generalized Additive Models
Version 0.1.7
Date 2013-5-7
Author Raymond K. W. Wong, Fang Yao and Thomas C. M. Lee
Maintainer Raymond K. W. Wong <raymondkww.dev@gmail.com>
Description This package provides robust estimation for generalized additive models. It implements a fast and stable algorithm in Wong, Yao and Lee (2013). The implementation also contains three automatic selection methods for smoothing parameter. They are designed to be robust to outliers. For more details, see Wong, Yao and Lee (2013).
License GPL (>= 2)
Depends Rcpp (>= 0.9.13), RcppArmadillo (>= 0.3.4.4) , mgcv (>= 1.7-20), robustbase (>= 0.9-3)
LinkingTo Rcpp, RcppArmadillo
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-01-02 19:18:27

R topics documented:

  robustgam-package ..................................................... 2
  cplot.robustgam ..................................................... 2
  pred.robustgam ..................................................... 4
  robustgam ............................................................ 6
  robustgam.CV ......................................................... 8
  robustgam.GIC ........................................................ 11
  robustgam.GIC.optim .................................................. 14

Index 18
Description

This package provides robust estimation for generalized additive models. It implements a fast and stable algorithm in Wong, Yao and Lee (2013). The implementation also contains three automatic selection methods for smoothing parameter. They are designed to be robust to outliers. For more details, see Wong, Yao and Lee (2013).

Details

Package: robustgam
Type: Package
Version: 0.1.5
Date: 2013-1-6
License: GPL (>= 2)

Author(s)

Raymond K. W. Wong Maintainer: Raymond K. W. Wong <raymondkww.dev@gmail.com>

References


Examples

# see example in function robustgam

cplot.robustgam

Description

This function plots the component smooth functions.

Usage

cplot.robustgam(fit, ranges, len=100)
Arguments

fit  
the robustgam fit
ranges  
matrix with each column containing the range of predictor in each plot
len  
grid size for the plot; it is used for all component smooth functions

Author(s)

Raymond K. W. Wong <raymondkww.dev@gmail.com>

References


See Also

robustgam.GIC, robustgam.GIC.optim, robustgam.CV.pred.robustgam

Examples

# load library
library(robustgam)

test.fun <- function(x, ...) {
  return(2*sin(2*pi*(1-x)^2))
}

# some setting
set.seed(1234)
true.family <- poisson()
out.prop <- 0.05
n <- 100

# generating dataset for poisson case
x <- runif(n)
x <- x[order(x)]
true.eta <- test.fun(x)
true.mu <- true.family$linkinv(test.fun(x))
y <- rpois(n, true.mu) # for poisson case

# create outlier for poisson case
out.n <- trunc(n*out.prop)
out.list <- sample(1:n, out.n, replace=FALSE)
y[out.list] <- round(y[out.list]*runif(out.n,min=3,max=5)^((sample(c(-1,1),out.n,TRUE))))

# robust GAM fit
robustfit <- robustgam(x, y, family=true.family, p=3, c=1.6, sp=0.000143514, show.msg=FALSE, smooth.basis='tp')
## the smoothing parameter is selected by the RBIC, the command is:
# robustfit.gic <- robustgam.GIC.optim(x, y, family=true.family, p=3, c=1.6, show.msg=FALSE,
# count.lim=400, smooth.basis='tp', lsp.initial=log(1e-2), lsp.min=-15, lsp.max=10,
# gic.constant=log(n), method="L-BFGS-B"; robustfit <- robustfit.gic$optim.fit

# ordinary GAM fit
nonrobustfit <- gam(y~s(x, bs="tp", m=3), family=true.family) # m = p for 'tp'

# prediction
x.new <- seq(range(x)[1], range(x)[2], len=1000)
robustfit.new <- pred.robustgam(robustfit, data.frame(x=x.new))$predict.values
nonrobustfit.new <- as.vector(predict.gam(nonrobustfit, data.frame(x=x.new), type="response"))

# plot
plot(x, y)
lines(x.new, true.family$linkinv(test.fun(x.new)), col="blue")
lines(x.new, robustfit.new, col="red")
lines(x.new, nonrobustfit.new, col="green")
legend(0.6, 23, c("true mu", "robust fit", "nonrobust fit"), col=c("blue","red","green"), lty=c(1,1,1))

# plot component smooth function
cplot.robustgam(robustfit, ranges=range(x))

---

**pred.robustgam**

**Prediction method for robustgam**

**Description**

A prediction function for output of `robustgam`.

**Usage**

```r
pred.robustgam(fit, data, type="response")
```

**Arguments**

- `fit`  
  fit object of `robustgam`

- `data`  
  a data.frame object. Call the variable `X` if there is only one covariate. Otherwise, call the first variable `X1`, the second one `X2`, and so on.

- `type`  
  type of output

**Value**

- `predict.comp`  
  a matrix containing the individual additive components for each covariates.

- `predict.values`  
  the type of output required by `type`. For example, if `type="response"`, the predicted values of the data is outputed.
Author(s)

Raymond K. W. Wong <raymondkww.dev@gmail.com>

References


See Also

`robustgam`

Examples

```r
# load library
library(robustgam)

# test function
test.fun <- function(x, ...) {
  return(2*sin(2*pi*(1-x)^2))
}

# some setting
set.seed(1234)
ture.family <- poisson()
out.prop <- 0.05
n <- 100

# generating dataset for poisson case
x <- runif(n)
x <- x[order(x)]
ture.eta <- test.fun(x)
ture.mu <- ture.family$linkinv(test.fun(x))
y <- rpois(n, ture.mu) # for poisson case

# create outlier for poisson case
out.n <- trunc(n*out.prop)
out.list <- sample(1:n, out.n, replace=FALSE)
y[out.list] <- round(y[out.list]*runif(out.n,min=3,max=5)^sample(c(-1,1),out.n,TRUE))

# robust GAM fit
robustfit <- robustgam(x, y, family=ture.family, p=3, c=1.6, sp=0.00143514, show.msg=FALSE, smooth.basis='tp')

# ordinary GAM fit
nonrobustfit <- gam(y~s(x, bs="tp", m=3),family=ture.family) # m = p for 'tp'

# prediction
x.new <- seq(range(x)[1], range(x)[2], len=1000)
robustfit.new <- predict(robustgam(robustfit, data.frame(X=x.new))$predict.values
nonrobustfit.new <- as.vector(predict.gam(nonrobustfit, data.frame(x=x.new), type="response"))
```
robustgam

Pseudo Data Algorithm for Robust Estimation of GAMs

Description

This function implements a fast and stable algorithm developed in Wong, Yao and Lee (2013) for robust estimation of generalized additive models. Currently, this implementation only covers binomial and poisson distributions. This function does not choose smoothing parameter by itself and the user has to specify it manually. For implementation with automatic smoothing parameter selection, see robustgam.GIC, robustgam.GIC.optim, robustgam.CV. For prediction, see pred.robustgam.

Usage

robustgam(x, y, family, p=3, K=30, c=1.345, sp=-1, show.msg=FALSE, countlim=200, w.countlim=50, smoothbasis="tp", wx=FALSE)

Arguments

X a vector or a matrix (each covariate form a column) of covariates
y a vector of responses
family A family object specifying the distribution and the link function. See glm and family.
p order of the basis. It depends on the option of smooth.basis.
K number of knots of the basis; dependent on the option of smooth.basis.
c tuning parameter for Huber function; a smaller value of c corresponds to a more robust fit. It is recommended to set as 1.2 and 1.6 for binomial and poisson distribution respectively.
sp a vector of smoothing parameter. If only one value is specified, it will be used for all smoothing parameters.
show.msg If show.msg=T, progress is displayed.
countlim maximum number of iterations of the whole algorithm
w.countlim maximum number of updates on the weight. It corresponds to zeta in Wong, Yao and Lee (2013)
smooth.basis the specification of basis. Four choices are available: "tp" = thin plate regression spline, "cr" = cubic regression spline, "ps" = P-splines, "tr" = truncated power spline. For more details, see smooth.construct.
wx If wx=T, robust weight on the covariates are applied. For details, see Real Data Example in Wong, Yao and Lee (2013)
Value

- `fitted.values`: fitted values
- `initial.fitted`: the starting values of the algorithm
- `beta`: estimated coefficients (corresponding to the basis)
- `B`: the basis: fitted linear estimator = \( B \times beta \)
- `sD`: for internal use
- `basis`: the smooth construct object. For more details, see `smooth.construct`
- `converge`: If `converge=T`, the algorithm converged.
- `w`: for internal use
- `family`: the family object
- `wx`: Indicate whether robust weight on covariates is applied
- `beta.fit`: for internal use

Author(s)

Raymond K. W. Wong <raymondkww.dev@gmail.com>

References


See Also

- `robustgam.GIC`, `robustgam.GIC.optim`, `robustgam.CV`, `pred.robustgam`

Examples

```r
# load library
library(robustgam)

# test function
test.fun <- function(x, ...) {
  return(2*sin(2*pi*(1-x)^2))
}

# some setting
set.seed(1234)
true.family <- poisson()
out.prop <- 0.05
n <- 100

# generating dataset for poisson case
x <- runif(n)
x <- x[order(x)]
true.eta <- test.fun(x)
true.mu <- true.family$linkinv(test.fun(x))
```
y <- rpois(n, true.mu)  # for poisson case

# create outlier for poisson case
out.n <- trunc(n*out.prop)
out.list <- sample(1:n, out.n, replace=FALSE)
y[out.list] <- round(y[out.list]*runif(out.n, min=3, max=5)*(sample(c(-1,1), out.n, TRUE)))

# robust GAM fit
robustfit <- robustgam(x, y, family=true.family, p=3, c=1.6, sp=0.000143514, show.msg=FALSE,
                       smooth.basis='tp')
## the smoothing parameter is selected by the RBIC, the command is:
## robustfit.gic <- robustgam.GIC.optim(x, y, family=true.family, p=3, c=1.6, show.msg=FALSE,
##  count.lim=400, smooth.basis='tp', lsp.initial=log(1e-2) ,lsp.min=-15, lsp.max=10,
##  gic.constant=log(n), method="L-BFGS-B"); robustfit <- robustfit.gic$optim.fit

# ordinary GAM fit
nonrobustfit <- gam(y~s(x, bs="tp", m=3),family=true.family)  # m = p for 'tp'

# prediction
x.new <- seq(range(x)[1], range(x)[2], len=1000)
robustfit.new <- pred.robustgam(robustfit, data.frame(X=x.new))$predict.values
nonrobustfit.new <- as.vector(predict.gam(nonrobustfit, data.frame(x=x.new), type="response"))

# plot
plot(x, y)
lines(x.new, true.family$linkinv(test.fun(x.new)), col="blue")
lines(x.new, robustfit.new, col="red")
lines(x.new, nonrobustfit.new, col="green")
legend(0.6, 20, c("true mu", "robust fit", "nonrobust fit"), col=c("blue","red","green"),
       lty=c(1,1,1))

---

robustgam.CV  
Smoothing parameter selection by robust cross validation

Description

This function combines the robustgam with automatic smoothing parameter selection. The smoothing parameter is selected through robust cross validation criterion described in Wong, Yao and Lee (2013). The criterion is designed to be robust to outliers. This function uses grid search to find the smoothing parameter that minimizes the criterion.

Usage

robustgam.CV(X, y, family, p=3, K=30, c=1.345, show.msg=FALSE, count.lim=200,
            w.count.lim=50, smooth.basis="tp", wx=FALSE, sp.min=1e-7, sp.max=1e-3,
            len=50, show.msg.2=TRUE, ngroup=length(y), seed=12345)
robustgam.CV

Arguments

- **X**: a vector or a matrix (each covariate form a column) of covariates
- **y**: a vector of responses
- **family**: A family object specifying the distribution and the link function. See `glm` and `family`.
- **p**: order of the basis. It depends on the option of smooth.basis.
- **K**: number of knots of the basis; dependent on the option of smooth.basis.
- **c**: tuning parameter for Huber function; a smaller value of c corresponds to a more robust fit. It is recommended to set as 1.2 and 1.6 for binomial and poisson distribution respectively.
- **show.msg** If `show.msg=T`, progress of robustgam is displayed.
- **count.lim**: maximum number of iterations of the whole algorithm
- **w.count.lim**: maximum number of updates on the weight. It corresponds to zeta in Wong, Yao and Lee (2013)
- **smooth.basis**: the specification of basis. Four choices are available: "tp" = thin plate regression spline, "cr" = cubic regression spline, "ps" = P-splines, "tr" = truncated power spline. For more details, see `smooth.construct`.
- **wx**: If `wx=T`, robust weight on the covariates are applied. For details, see Real Data Example in Wong, Yao and Lee (2013)
- **sp.min**: A vector of minimum values of the searching range for smoothing parameters. If only one value is specified, it will be used for all smoothing parameters.
- **sp.max**: A vector of maximum values of the searching range for smoothing parameters. If only one value is specified, it will be used for all smoothing parameters.
- **len**: A vector of grid sizes. If only one value is specified, it will be used for all smoothing parameters.
- **show.msg.2**: If `show.msg.2=T`, progress of the grid search is displayed.
- **ngroup**: number of group used in the cross validation. If `ngroup=length(y)`, full cross validation is implemented. If `ngroup=2`, two-fold cross validation is implemented.
- **seed**: The seed for random generator used in generating partitions.

Value

- **fitted.values**: fitted values (of the optimum fit)
- **initial.fitted**: the starting values of the algorithm (of the optimum fit)
- **beta**: estimated coefficients (corresponding to the basis) (of the optimum fit)
- **optim.index**: the index of the optimum fit
- **optim.index2**: the index of the optimum fit in another representation:
  ```r
  optim.ndex2=as.arrayInd(optim.index,len)
  ```
- **optim.criterion**: the optimum value of robust cross validation criterion
optim.sp  the optimum value of the smoothing parameter
criteria  the values of criteria for all fits during grid search
sp        the grid of smoothing parameter
optim.fit the robustgam fit object of the optimum fit. It is handy for applying the prediction method.

Author(s)
Raymond K. W. Wong <raymondkww.dev@gmail.com>

References

See Also
robustgam.GIC, robustgam.GIC.optim, robustgam.CV, pred.robustgam

Examples

```r
# load library
library(robustgam)

# test function
test.fun <- function(x, ...) {
  return(2*sin(2*pi*(1-x)^2))
}

# some setting
set.seed(1234)
ture.family <- poisson()
out.prop <- 0.05
n <- 100

# generating dataset for poisson case
x <- runif(n)
x <- x[order(x)]
ture.eta <- test.fun(x)
ture.mu <- ture.family$linkinv(test.fun(x))
y <- rpois(n, ture.mu)  # for poisson case

# create outlier for poisson case
out.n <- trunc(n*out.prop)
out.list <- sample(1:n, out.n, replace=FALSE)
y[out.list] <- round(y[out.list]*runif(out.n,min=3,max=5)^*(sample(c(-1,1),out.n,TRUE)))

# Not run:

# robust GAM fit
robustfit.gic <- robustgam.CV(x, y, family=ture.family, p=3, c=1.6, show.msg=FALSE)
```


robustgam.GIC

Smoothing parameter selection by GIC (grid search)

Description

This function combines the robustgam with automatic smoothing parameter selection. The smoothing parameter is selected through generalized information criterion (GIC) described in Wong, Yao and Lee (2013). The GIC is designed to be robust to outliers. There are two criteria for user to choose from: Robust AIC and Robust BIC. The robust BIC usually gives smoother surface than robust AIC. This function uses grid search to find the smoothing parameter that minimizes the criterion.

Usage

```r
robustgam.GIC(X, y, family, p=3, K=30, c=1.345, show.msg=FALSE, count.lim=200,
              w.count.lim=50, smooth.basis="tp", wx=FALSE, sp.min=1e-7, sp.max=1e-3,
              len=50, show.msg.2=TRUE, gic.constant=log(length(y)))
```

Arguments

- `X`  
  a vector or a matrix (each covariate form a column) of covariates
- `y`  
  a vector of responses
- `family`  
  A family object specifying the distribution and the link function. See `glm` and `family`.
- `p`  
  order of the basis. It depends on the option of smooth.basis.
- `K`  
  number of knots of the basis; dependent on the option of smooth.basis.
c: Tuning parameter for Huber function; a smaller value of c corresponds to a more robust fit. It is recommended to set as 1.2 and 1.6 for binomial and poisson distribution respectively.

show.msg: If show.msg=T, progress of robustgam is displayed.

count.lim: Maximum number of iterations of the whole algorithm.

w.count.lim: Maximum number of updates on the weight. It corresponds to zeta in Wong, Yao and Lee (2013).

smooth.basis: The specification of basis. Four choices are available: "tp" = thin plate regression spline, "cr" = cubic regression spline, "ps" = P-splines, "tr" = truncated power spline. For more details, see smooth.construct.

wx: If wx=T, robust weight on the covariates are applied. For details, see Real Data Example in Wong, Yao and Lee (2013).

sp.min: A vector of minimum values of the searching range for smoothing parameters. If only one value is specified, it will be used for all smoothing parameters.

sp.max: A vector of maximum values of the searching range for smoothing parameters. If only one value is specified, it will be used for all smoothing parameters.

ten: A vector of grid sizes. If only one value is specified, it will be used for all smoothing parameters.

show.msg.2: If show.msg.2=T, progress of the grid search is displayed.

gic.constant: If gic.constant=log(length(y)), robust BIC is used. If gic.constant=2, robust AIC is used.

Value:

fitted.values: Fitted values (of the optimum fit)

initial.fitted: The starting values of the algorithm (of the optimum fit)

beta: Estimated coefficients (corresponding to the basis) (of the optimum fit)

optim.index: The index of the optimum fit

optim.index2: The index of the optimum fit in another representation:

optim.index2=arrayInd(optim.index,len)

optim.gic: The optimum value of robust AIC or robust BIC

optim.sp: The optimum value of the smoothing parameter

fit.list: A list object containing all fits during the grid search

gic: The values of GIC for all fits during grid search

gic.comp1: For internal use

gic.comp2: For internal use

sp: The grid of smoothing parameter

gic.constant: The gic.constant specified in the input

optim.fit: The robustgam fit object of the optimum fit. It is handy for applying the prediction method.
Author(s)

Raymond K. W. Wong <raymondkww.dev@gmail.com>

References


See Also

robustgam.GIC, robustgam.GIC.optim, robustgam.CV, pred.robustgam

Examples

```r
# load library
library(robustgam)

# test function
test.fun <- function(x, ...) {
  return(2*x*sin(2*pi*(1-x)^2))
}

# some setting
set.seed(1234)
true.family <- poisson()
out.prop <- 0.05
n <- 100

# generating dataset for poisson case
x <- runif(n)
x <- x[order(x)]
true.eta <- test.fun(x)
true.mu <- true.family$linkinv(test.fun(x))
y <- rpois(n, true.mu) # for poisson case

# create outlier for poisson case
out.n <- trunc(n*out.prop)
out.list <- sample(1:n, out.n, replace=FALSE)
y[out.list] <- round(y[out.list]*runif(out.n,min=3,max=5)^sample(c(-1,1),out.n,TRUE))

## Not run:

# robust GAM fit
robustfit.gic <- robustgam.GIC(x, y, family=true.family, p=3, c=1.6, show.msg=FALSE, count.lim=400, smooth.basis='tp'); robustfit <- robustfit.gic$optim.fit

# ordinary GAM fit
nonrobustfit <- gam(y~s(x, bs="tp", m=3),family=true.family) # m = p for 'tp'

# prediction
x.new <- seq(range(x)[1], range(x)[2], len=1000)
robustfit.new <- pred.robustgam(robustfit, data.frame(X=x.new))$predict.values
```
robustgam.GIC.optim

Description
This function is the same as robustgam.GIC, except that the R internal optimization function optim is used to find the smoothing parameter that minimizes the RAIC or RBIC criterion.

Usage
robustgam.GIC.optim(X, y, family, p=3, K=30, c=1.345, show.msg=FALSE, count.lim=200, w.count.lim=50, smooth.basis=“tp”, wx=FALSE, lsp.initial=log(1e-4), lsp.min=-20, lsp.max=10, gic.constant=log(length(y)), method="L-BFGS-B", optim.control=list(trace=1))

Arguments
X a vector or a matrix (each covariate form a column) of covariates
y a vector of responses
family A family object specifying the distribution and the link function. See glm and family.
p order of the basis. It depends on the option of smooth.basis.
K number of knots of the basis; dependent on the option of smooth.basis.
c tuning parameter for Huber function; a smaller value of c corresponds to a more robust fit. It is recommended to set as 1.2 and 1.6 for binomial and poisson distribution respectively.
show.msg If show.msg=T, progress of robustgam is displayed.
count.lim maximum number of iterations of the whole algorithm
w.count.lim maximum number of updates on the weight. It corresponds to zeta in Wong, Yao and Lee (2013)
smooth.basis the specification of basis. Four choices are available: “tp” = thin plate regression spline, “cr” = cubic regression spline, “ps” = P-splines, “tr” = truncated power spline. For more details, see smooth.construct.
If \( wx = \text{T} \), robust weight on the covariates are applied. For details, see Real Data Example in Wong, Yao and Lee (2013).

**lsp.initial** A vector of initial values of the \( \log \) of smoothing parameters used to start the optimization algorithm.

**lsp.min** a vector of minimum values of the searching range for the \( \log \) of smoothing parameters. If only one value is specified, it will be used for all smoothing parameters.

**lsp.max** a vector of maximum values of the searching range for the \( \log \) of smoothing parameters. If only one value is specified, it will be used for all smoothing parameters.

**gic.constant** If \( \text{gic.constant} = \log(\text{length}(y)) \), robust BIC is used. If \( \text{gic.constant} = 2 \), robust AIC is used.

**method** method of optimization. For more details, see \texttt{optim}.

**optim.control** setting for \texttt{optim}. For more details, see \texttt{optim}.

---

**Value**

- **fitted.values** fitted values (of the optimum fit)
- **beta** estimated coefficients (corresponding to the basis) (of the optimum fit)
- **beta.fit** for internal use
- **gic** the optimum value of robust AIC or robust BIC
- **sp** the optimum value of the smoothing parameter
- **gic.optim** the output of \texttt{optim}
- **w** for internal use
- **gic.constant** the \texttt{gic.constant} specified in the input
- **optim.fit** the \texttt{robustgam} fit object of the optimum fit. It is handy for applying the prediction method.

---

**Author(s)**

Raymond K. W. Wong <raymondkww.dev@gmail.com>

**References**


**See Also**

\texttt{robustgam.GIC}, \texttt{robustgam.GIC.optim}, \texttt{robustgam.CV}, \texttt{pred.robustgam}
Examples

```r
# load library
library(robustgam)

# test function
test.fun <- function(x, ...) {
  return(2*sin(2*pi*(1-x)^2))
}

# some setting
set.seed(1234)
true.family <- poisson()
out.prop <- 0.05
n <- 100

# generating dataset for poisson case
x <- runif(n)
x <- x[order(x)]
true.eta <- test.fun(x)
true.mu <- true.family$linkinv(test.fun(x))
y <- rpois(n, true.mu)  # for poisson case

# create outlier for poisson case
out.n <- trunc(n*out.prop)
out.list <- sample(1:n, out.n, replace=FALSE)
y[out.list] <- round(y[out.list]*runif(out.n,min=3,max=5)^((sample(c(-1,1),out.n,TRUE)))

## Not run:

# robust GAM fit
robustfit.gic <- robustgam.GIC.optim(x, y, family=true.family, p=3, c=1.6, show.msg=FALSE,
count.lim=400, smooth.basis='tp', lsp.initial=log(1e-2), lsp.min=-15, lsp.max=10,
gic.constant=log(n), method="L-BFGS-B"); robustfit <- robustfit.gic$optim.fit

# ordinary GAM fit
nonrobustfit <- gam(y~s(x, bs="tp", m=3),family=true.family)  # m = p for 'tp'

# prediction
x.new <- seq(range(x)[1], range(x)[2], len=1000)
robustfit.new <- pred.robustgam(robustfit, data.frame(X=x.new))$predict.values
nonrobustfit.new <- as.vector(predict.gam(nonrobustfit, data.frame(x=x.new), type="response"))

# plot
plot(x, y)
lines(x.new, true.family$linkinv(test.fun(x.new)), col="blue")
lines(x.new, robustfit.new, col="red")
lines(x.new, nonrobustfit.new, col="green")
legend(0.6, 23, c("true mu", "robust fit", "nonrobust fit"), col=c("blue","red","green"),
       lty=c(1,1,1))
```
## End (Not run)
Index

*Topic Bounded score function, Generalized information criterion, Generalized linear model, Robust estimating equation, Robust quasi-likelihood, Smoothing parameter selection

robustgam-package, 2

cplot.robustgam, 2

family, 6, 9, 11, 14

glm, 6, 9, 11, 14

optim, 14, 15

pred.robustgam, 3, 4, 6, 7, 10, 13, 15

robustgam, 4, 5, 6, 8, 11

robustgam-package, 2

robustgam.CV, 3, 6, 7, 8, 10, 13, 15

robustgam.GIC, 3, 6, 7, 10, 11, 13–15

robustgam.GIC.optim, 3, 6, 7, 10, 13, 14, 15

smooth.construct, 6, 7, 9, 12, 14