Package ‘scoop’

September 16, 2011

Version 0.2-1

Date 2011-xx-xx

Title Sparse cooperative regression

Author Julien Chiquet

Maintainer Julien Chiquet <julien.chiquet@genopole.cnrs.fr>

Depends MASS, methods

URL http://stat.genopole.cnrs.fr/logiciels/scoop

Description This package fits coop-Lasso, group-Lasso, tree-group
Lasso and Lasso solution paths for linear regression and
logistic regression. The cooperative-Lasso (in short
coop-Lasso) may be viewed as a modification of the group-Lasso
penalty that promotes sign coherence and that allows zeros within groups.

License GPL (>= 2)

Archs i386, x86_64

R topics documented:

scoop-package 2
basal 2
crossval,scoopfit-methods 3
cvscoop-class 4
plot,cvscoop-methods 5
plot,scoopfit-methods 6
scoop 8
scoopfit-class 10
selection,scoopfit-methods 12

Index **14**

`scoop-package` *Sparse cooperative regression*

Description

This package fits coop-Lasso and group-Lasso paths of solutions for linear and logistic regressions. The cooperative-Lasso (in short *coop-Lasso*, as introduced in Chiquet et al., 2010, 2011) may be viewed as a modification of the group-Lasso penalty that promotes sign coherence and allows zeros within groups.

The main function to fit a model is `scoop`. It produces an object of class `scoop` for which `predict`, `fitted`, `deviance`, `residuals`, `print` and `plot` methods exist.

Two other methods are included for model selection purpose (that is, to choose an appropriate amount of penalization): the `selection` function estimates AIC and BIC criteria and relies on an approximation of the degrees of freedom of a coop-Lasso fit. The `crossval` function performs cross-validation and produces an object of class `cvscoop` for which a `plot` method exists.

Note that this is an early release for peer-review purpose.

Demo available

`demo(basal_tumor)` Example on the `basal` data set that compares logistic coop-Lasso fit and logistic group-Lasso fit on a selected number of probe sets. Cross-validation is performed and selected genes depicted.

Author(s)

Julien Chiquet, <julien.chiquet@genopole.cnrs.fr> (also maintainer).

References

Julien Chiquet, Yves Grandvalet and Camille Charbonnier (2011 - preprint). Sparsity with sign-coherent groups of variables via the cooperative-Lasso.

Julien Chiquet, Yves Grandvalet and Christophe Ambroise (2010). Inferring multiple graphical structures, Statistics and Computing, http://dx.doi.org/10.1007/s11222-010-9191-2.

`basal` *Microarray data set on breast cancer (basal tumors)*

Description

This gene expression data set is freely available, coming from Hess *et al*'s paper. Patients were treated with chemotherapy prior to surgery. Patient response to the treatment can be classified as either a pathologic complete response (pCR) or residual disease (not-pCR). The original data set concerns one hundred thirty-three patients with stage I–III breast cancer.

Following Jeanmougin et al. (2011), we restrict to the twenty-nine basal tumors from the available samples, divided into 15 pCR and 14 not-pCR. This particular subtype of breast cancer, harboring homogeneous clinical and pathologic features, shows highly variable response to chemotherapy. For illustration purpose, we only keep the first ten most differentiated probe sets, which exactly match ten genes. We also integrate all the probe sets related to these ten genes, regardless of their p-values. The `basal` data ends up with 381 variables (the probe sets) clustered into 172 groups (the genes) for a total of 29 samples (the patients).

Usage

```
data(basal)
```

Format

The four objects are created when loading `data(basal)`:

`x` matrix with 381 columns and 29 rows. The kth row gives the expression levels of the 381 selected probe sets related to the kth patient.

`y` binary vector of size 29 indicating the patient status (`"pcr"` and `"not"`).

`group` vector of size 381 indicating the group belonging (that is, the gene each probe set is related to).

`genes` vector of size 172 with the gene names.

References

Marine Jeanmougin, Mickael Guedj, Christophe Ambroise (preprint, 2011) *Defining a robust biological prior from Pathway Analysis to drive Network Inference*, `http://arxiv.org/abs/1101.3493`.

K.R. Hess, K. Anderson, W.F. Symmans, V. Valero, N. Ibrahim, J.A. Mejia, D. Booser, R.L. Theriault, U. Buzdar, P.J. Dempsey, R. Rouzier, N. Sneige, J.S. Ross, T. Vidaurre, H.L. Gomez, G.N. Hortobagyi, and L. Pustzai (2006). Pharmacogenomic predictor of sensitivity to preoperative chemotherapy with Paclitaxel and Fluorouracil, Doxorubicin, and Cyclophosphamide in breast cancer, *Journal of Clinical Oncology*, vol. 24(26), pp. 4236–4244.

Examples

```
data(basal)
cat("\nmight take about half a minute...")
grplasso <- group.lasso(x, y, group, family="binomial", lambda.min=1e-4)
coolasso <- coop.lasso(x, y, group, family="binomial", lambda.min=1e-4)
par(mfrow=c(1,2))
plot(grplasso, yvar="group")
plot(coolasso, yvar="group")
```

`crossval,scoopfit-methods`
Cross-validation of a scoop model

Description

Function that computes K-fold cross-validated error of a `scoop` fit.

Usage

```
## S4 method for signature 'scoopfit'
crossval(object,
          K         = 10,
          error     = "deviance",
          threshold = 0.5,
          folds     = NULL,
          verbose   = TRUE)
```

Arguments

`object`	output of a `scoop` run (must be an object of class `scoopfit`).
`K`	integer that indicates the number of folds.
`error`	string that indicates the loss to use for cross-validation, either `"deviance"` or `"classification"`. The latter is only available for the binomial family. Default is `"deviance"`.
`threshold`	number between 0 and 1 for the decision threshold when `"classification"` error is required.
`folds`	list of `K` vectors that describes the folds to use for the cross-validation. If `NULL`, they are randomly sampled. Default is `NULL`.
`verbose`	logical; indicates if the progression (the current fold number) should be displayed. Default is `TRUE`.

Value

Returns an object of class `cvscoop` for which a `plot` method is available.

Author(s)

Julien Chiquet, inspired by the function `cv.glmnet` of the **glmnet** package by J. Friedman, T. Hastie and R. Tibshirani.

See Also

See also as `plot.cvscoop`, `scoop`, `cvscoop`, `scoopfit`.

Examples

```
data(basal)
set.seed(47)
coolasso <- coop.lasso(x, y, group, family="binomial", n.lambda=50)
out.cv <- crossval(coolasso, K=5)
plot(out.cv)
```

`cvscoop-class` *Class "cvscoop"*

Description

Class of object returned by the `crossval` method.

Objects from the Class

Objects can be created by calls to `crossval`.

Slots

`lambda`: vector of penalty levels for which each cross-validation has been performed.

`lambda.min`: level of penalty that minimizes the mean cross-validated error.

`lambda.1se`: largest level of penalty such has the cross-validated error is within 1 standard error of the minimum.

`cv.mean`: vector with the same length as `lambda` containing the mean cross-validated error.

`cv.error`: vector with the same length as `lambda` containing the estimated standard deviation of `cv.mean`.

`folds`: list of `K` vector indicating the folds used for cross-validation.

`beta.min`: vector of parameters corresponding to `lambda.min`.

`beta.1se`: vector of parameters corresponding to `lambda.1se`.

Methods

A `plot` method is available and documented.

Author(s)

Julien Chiquet

See Also

See also `plot.cvscoop`.

Examples

```
showClass("cvscoop")
```

`plot,cvscoop-methods` *Plot method for cross validated error of a scoop model*

Description

Produce a plot of the cross validated error of a `scoop` model.

Usage

```
## S4 method for signature 'cvscoop'
plot(x, y,
        log.scale = TRUE,
        xlab = ifelse(log.scale, "lambda (log scale)", "lambda"),
        ylab = "Mean CV error", ...)
```

Arguments

`x`	output of a `crossval` run (must be of class `cvscoop`).
`y`	used for S4 compatibility
`xlab`	title for the x axis. If `NULL`, will be set to `"lambda"` or `"lambda (log-scale)"` according to `log.scale`.
`ylab`	title for the y axis
`log.scale`	logical; indicates if a log-scale should be used.
`...`	additional arguments for generic plot.

Author(s)

Julien Chiquet

See Also

See also `cvscoop`, `crossval.scoop`.

Examples

```
data(basal)
set.seed(47)
cat("\nmight take about half a minute...")
coolasso <- coop.lasso(x, y, group, family="binomial", n.lambda=50)
out.cv <- crossval(coolasso, K=5)
plot(out.cv)
```

`plot,scoopfit-methods` *Plot method for scoop object*

Description

Produce a plot of the solution path of a `scoop` fit.

Usage

```
## S4 method for signature 'scoopfit'
plot(x, y,
            xvar = "lambda", yvar="coefficients",
            main = paste(x@penalty," path (", yvar, ")", sep=""),
            crit = NULL, log.scale = TRUE, labels = NULL,
            col = c(1+switch(yvar,
              group = c(1:nlevels(as.factor(x@group))),
              c(x@group))),
            lty = switch(yvar,
              group = c(1:nlevels(as.factor(x@group))),
              c(x@group)), ...)
```

Arguments

`x`	output of a `scoop` fit (must be of class `scoopfit`).
`y`	used for S4 compatibility.
`xvar`	variable to plot on the X-axis: either `"lambda"`, `"fraction"` or `"df"`. Default is set to `"lambda"`.
`yvar`	variable to plot on the Y-axis: either `"coefficients"` or `"group"` for plotting the group-norm. Default is set to `"coefficients"`.
`main`	main title. Default is set to the model name followed by what is on the Y-axis.
`crit`	vector depicting the chosen criterion with the same number of entries as the number of `lambda` values in `x` (typically, BIC, AIC, CV-error or test-error). If specified, a vertical line is plotted to represent the value of `lambda` which minimizes this criterion.
`log.scale`	logical; indicates if a log-scale should be used when `xvar="lambda"`.
`labels`	vector indicating the names associated to the plotted variables (either the names of the coefficients or the names of the groups, depending on `yvar`). When specified, the names appear on the Y-axis in order to identify each variable in the solution path. Remind that the intercept does not count. Default is `NULL`.
`col`	specification for the plotting color. Default is to use the same color within a group.
`lty`	specification for the line type. Default is to use the same type within a group.
`...`	additional arguments for generic `plot`.

Author(s)

Julien Chiquet

See Also

See also `scoop`, `scoopfit`.

Examples

```
set.seed(87)
## true parameters (including intercept)
beta <- c(2, 0.8, 1.1, 0, -1.5, 0.3, -1.0)
## two groups with three components
group <- c(1, 1, 1, 2, 2, 2)
## Labels for coefficients and groups
labels.coef <- c("v11","v12","v13","v21","v22", "v23")
labels.grp  <- c("g1","g2")
## random design and additive noise with sd = 2
n <- 60
p <- 6
x <- matrix(rnorm(p * n), nrow = n)
y <- cbind(1, x) %*% beta + rnorm(n,0,2)
## fit coop-lasso and look for BIC selection capability
coolasso <- coop.lasso(x, y, group, lambda.min=0.1)
scoo     <- selection(coolasso)
## plot the coefficient path with chosen model
par(mfrow=c(1,2))
plot(coolasso, crit=scoo$BIC, labels=labels.coef)
plot(coolasso, crit=scoo$BIC, yvar="group", labels=labels.grp)
```

`scoop` *Function to fit a penalized regression problem with various grouping effects.*

Description

Fit a penalized regression problem with possible various grouping effects. Implements variants such as Lasso, group-Lasso, coop-Lasso, sparse group-Lasso, sparse coop-Lasso, tree group-Lasso and tree coop-Lasso penalties.

`lasso`, `group.lasso`, `coop.lasso` and `sparse.group.lasso` are aliases of `scoop` for which the type of penalty does not need to be specified.

Usage

```
lasso(x, y, family = "gaussian", intercept = TRUE,
       normalize    = ifelse(family=="gaussian",TRUE,FALSE),
       wk           = rep(1,ncol(x)),
       lambda       = NULL, n.lambda = 100, lambda.min = 1e-2,
       verbose      = FALSE, eps = 1e-5, max.iter = 2*ncol(x),
       optim.method = ifelse(family=="gaussian","fista","bfgs"))

group.lasso(x, y, group, family = "gaussian", intercept = TRUE,
       normalize    = ifelse(family=="gaussian",TRUE,FALSE),
       wk           = sqrt(tabulate(group)),
       lambda       = NULL, n.lambda = 100, lambda.min = 1e-2,
       verbose      = FALSE, eps = 1e-5, max.iter = 2*ncol(x),
       optim.method = ifelse(family=="gaussian","fista","bfgs"))

coop.lasso(x, y, group, family = "gaussian", intercept = TRUE,
       normalize    = ifelse(family=="gaussian",TRUE,FALSE),
       wk           = sqrt(tabulate(group)),
       lambda       = NULL, n.lambda = 100, lambda.min = 1e-2,
       verbose      = FALSE, eps = 1e-5, max.iter = 2*ncol(x),
       optim.method = ifelse(family=="gaussian","fista","bfgs"))

sparse.group.lasso(x, y, group, family = "gaussian", intercept = TRUE,
       normalize    = ifelse(family=="gaussian",TRUE,FALSE),
       wk           = sqrt(tabulate(group)),
       lambda       = NULL, n.lambda = 100, lambda.min = 1e-2,
       verbose      = FALSE, eps = 1e-5, max.iter = 2*ncol(x))

sparse.coop.lasso(x, y, group, family = "gaussian", intercept = TRUE,
       normalize    = ifelse(family=="gaussian",TRUE,FALSE),
       wk           = sqrt(tabulate(group)),
       lambda       = NULL, n.lambda = 100, lambda.min = 1e-2,
       verbose      = FALSE, eps = 1e-5, max.iter = 2*ncol(x))

tree.group.lasso(x, y, group, family = "gaussian", intercept = TRUE,
       normalize    = ifelse(family=="gaussian",TRUE,FALSE),
       wk           = lapply(apply(group,1,tabulate),sqrt),
       lambda       = NULL, n.lambda = 100, lambda.min = 1e-2,
       verbose      = FALSE, eps = 1e-5, max.iter = 2*ncol(x))

tree.coop.lasso(x, y, group, family = "gaussian", intercept = TRUE,
       normalize    = ifelse(family=="gaussian",TRUE,FALSE),
       wk           = lapply(apply(group,1,tabulate),sqrt),
       lambda       = NULL, n.lambda = 100, lambda.min = 1e-2,
       verbose      = FALSE, eps = 1e-5, max.iter = 2*ncol(x))
```

Arguments

`x`	design matrix (do NOT include intercept).
`y`	response vector.
`group`	defines how variables should be grouped. Components sharing the same number define a group. Should be a vector for funciont `lasso`, `group.lasso`, `coop.lasso`, `sparse.group.lasso` and `sparse.coop.lasso`. Should be a matrix for `tree.group.lasso`, `tree.coop.lasso` with as much line as there are levels in the tree. See details.
`family`	response type. For now, either `"gaussian"` for linear regression or `"binomial"` for logistic regression. Default is `"gaussian"`.
`intercept`	logical; indicates if an intercept should be included in the model. Default is `TRUE`.
`normalize`	logical; indicates if variables should be normalized to have unit L2 norm before fitting. Default is `TRUE` for linear regression, `FALSE` for logistic regression.
`lambda`	vector of decreasing penalty levels. If `NULL` (the default), an appropriate vector will be generated with `n.lambda` entries, starting from a level of penalty where only the intercept is included, and then shrinked to `lambda.min`.
`n.lambda`	integer that indicates the number of values to put in the `lambda` vector. Ignored if `lambda` is provided.
`lambda.min`	minimal value of penalty that will be tried. Default is `1e-3`. Ignored if `lambda` is provided.
`wk`	an object with real postitive values that weight the penalty of each group of features. The default weights each feature according to the square root of the size of the group it belongs to. Should be a vector with `nlevels(group)` elements for `lasso`, `group.lasso`, `coop.lasso`, `sparse.group.lasso` and `sparse.coop.lasso`. Should be a list with as much entries as there are levels in the tree for `tree.group.lasso`, `tree.coop.lasso`. Each element in the list contains as many elements as groups at the present level.
`verbose`	logical; indicates verbose mode to display progression. Default is `FALSE`.
`eps`	tolerance used to stop the optimization algorithm. Default is `1e-4`.
`max.iter`	maximum number of iterations before the whole optimization algorithm stops (for a fixed value of lambda).
`optim.method`	The optimization method used to solve the underlying optimization problem on the currently activated set of variables. Can be `"bfgs"` (a BFGS quasi-Newton method with box constraints), `"ista"` or `"fista"` (proximal methods). Default is `"fista"` for linear regression and `"bfgs"` for logistic regression (the fastest in each category of problem). Only `"fista"` is available for `sparse.group.lasso`, `sparse.coop.lasso`, `tree.group.lasso` and `tree.coop.lasso`.

Details

The general strategy of the algorithm relies on maintaining a working set of variable, starting from a vector of zero and entering the features by groups of variables. The underlying optimization problem is solved only on the activated groups, thus solving small smooth problems with increasing size. The working set algorithm stops when the optimality conditions are met, up to a tolerance level defined by the first element of the `eps` argument. This also controls the convergence of the underlying, smooth optimization problem on the currently activated variable. Various methods are implemented to solve this latter problem, which is controlled by `optim.method`. The Lasso is implemented here and enjoys the tools of the **scoop** package, yet note that it is not as performant as in **lars** or **glmnet**.

Value

Returns an object of class `scoopfit` (see the documentation).

Author(s)

Julien Chiquet

See Also

See also `scoopfit`, `plot.scoop`, `selection` and `crossval`.

Examples

```
set.seed(87)
## true parameters (including intercept)
beta <- c(2, 0.8, 1.1, 0, -1.5, 0.3, -1.0)
## two groups with three components
group <- c(1, 1, 1, 2, 2, 2)
## random design and additive noise with sd = 2
n <- 60
p <- 6
x <- matrix(rnorm(p * n), nrow = n)
y <- cbind(1, x) %*% beta + rnorm(n,0,2)
## fit coop-lasso and look for BIC selection capability
coolasso <- coop.lasso(x, y, group, lambda.min=0.1)
scoo     <- selection(coolasso)
## plot the coefficient path with the chosen model
plot(coolasso, crit=scoo$BIC)
```

`scoopfit-class` *Class "scoopfit"*

Description

Class of object returned by the `scoop` fitting function.

Objects from the Class

Such objects can be created by calls to `scoop`, `lasso`, `group.lasso` and `coop.lasso`.

Slots

`coefficients`**:** matrix of coefficients with respect to the original input (even when normalized). The number of rows of `coefficients` corresponds the length of `lambda`.

`lambda`**:** vector of penalty levels for which the model has eventually been fitted.

`wk`**:** vector or lsit which defines how each group is weighted regarding the penalizer.

`group`**:** vector or matrix which defines the grouping of the variables.

`family`**:** response type used. For now, either `"gaussian"` for linear regression or `"binomial"` for logistic regression. Default is `"gaussian"`.

`penalty`**:** type of grouping effect used for the penalizer: either `"coop"`, `"group"` or `lasso`.

`monitoring`**:** list wich contains three vectors to monitor the optimization process: `it.active` gives the number of iterations in the active set algorithm for each value of lambda; `it.optim` gives the number of iterations for each call to the optimization function; `dual.gap` gives the duality gap reached for each lambda.

`x`**:** design matrix saved for further use.

`y`**:** response vector saved for further use.

`call`**:** original call to the `scoop.default` function.

Methods

This class comes with the usual `fitted(object, ...)`,`predict(object, newx=NULL, ...)`, `residuals(object, ...)`, `print(object, ...)` and `deviance(object, ...)` generic methods.

Three specific and documented methods are also available:

crossval cross-validates a `scoop` object

plot plots a `scoop` object

selection computes approximated BIC/AIC

Author(s)

Julien Chiquet

See Also

See Also `scoop`, `link{plot.scoop}`, `link{crossval}`, `link{selection}`.

Examples

```
showClass("scoopfit")
```

`selection,scoopfit-methods` *Model selection criteria for scoop models*

Description

Compute estimations of AIC and BIC of a `scoop` fit for model selection purpose. Only available for the Gaussian family (regular linear regression).

Usage

```
## S4 method for signature 'scoopfit'
selection(object, sigma2 = NULL)
```

Arguments

`object`	output of a `scoop` run (must be an object of class `scoopfit`)
`sigma2`	user defined estimator for the variance. If not specified, use the OLS estimate (only relevant when $n > p$). If $n < p$, the user should provide `sigma2` otherwise `BIC` and `AIC` will be `NULL`.

Details

The following expression are used to estimate the BIC (*Bayesian Information Criterion*)

$$\widetilde{\mathrm{BIC}}(\lambda) = \frac{\mathrm{RSS}(\lambda)}{\hat{\sigma}^2} + \log(n)\tilde{\mathrm{df}}(\lambda),$$

and the AIC (*Akaike Information Criterion*)

$$\widetilde{\mathrm{AIC}}(\lambda) = \frac{\mathrm{RSS}(\lambda)}{\hat{\sigma}^2} + 2\tilde{\mathrm{df}}(\lambda),$$

where $\tilde{\mathrm{df}}$ is an approximation of the degrees of freedom as described in Yuan and Lin (2006) and Chiquet et al (submitted).

Value

Returns a list comprising

`BIC`	vector containing the values of the BIC for the successive values of λ used along the `scoop` fit.
`beta.BIC`	vector of parameters chosen by minimizing BIC.
`lambda.BIC`	amount of penalization that minimizes BIC.
`AIC`	vector containing the values of the AIC for the succesive values of λ used along the `scoop` fit.
`beta.AIC`	vector of parameters chosen by minimizing AIC.
`lambda.AIC`	amount of penalization that minimizes BIC.
`sigma2`	estimated variance.
`df`	vector containing the degrees of freedom for the successive values of λ used along the `scoop` fit.
`non.zeros`	vector containing the number of non-zero coefficients for the successive values of λ used along the `scoop` fit.

Author(s)

Julien Chiquet

References

Julien Chiquet, Yves Grandvalet and Camille Charbonnier (2011 - preprint). Promotting sign coherence in groups of variables with the coop-Lasso, submitted.

M. Yuan and Y. Lin (2006). Model selection and estimation in regression with grouped variables, JRSS B, 8, Part 1, pp.49-67.

See Also

See also `scoop`, `scoopfit`.

Examples

```
## Not run:
set.seed(87)
## true parameters (including intercept)
beta <- c(2, 0.8, 1.1, 0, -1.5, 0.3, -1.0)
## two groups with three components
group <- c(1, 1, 1, 2, 2, 2)
## random design and additive noise with sd = 2
n <- 60
p <- 6
x <- matrix(rnorm(p * n), nrow = n)
y <- cbind(1, x) %*% beta + rnorm(n,0,2)
## fit coop-lasso and look for BIC selection capability
coolasso <- scoop(x, y, group, lambda.min=0.1)
scoo     <- selection(coolasso)
## BIC choice
scoo$beta.BIC
## End(Not run)
```

Index

∗Topic **classes**
  cvscoop-class, 4
  scoopfit-class, 10
∗Topic **datasets**
  basal, 2
∗Topic **hplot**
  plot,cvscoop-methods, 5
  plot,scoopfit-methods, 6
∗Topic **models**
  crossval,scoopfit-methods, 3
  scoop, 8
  selection,scoopfit-methods, 12
∗Topic **package**
  scoop-package, 2
∗Topic **regression**
  crossval,scoopfit-methods, 3
  scoop, 8
  selection,scoopfit-methods, 12

basal, 2

coop.lasso *(scoop)*, 8
crossval, *10*
crossval *(crossval,scoopfit-methods)*, 3
crossval,scoopfit-method *(crossval,scoopfit-methods)*, 3
crossval,scoopfit-methods, 3
crossval.scoop, *6*
crossval.scoop *(crossval,scoopfit-methods)*, 3
cvscoop, *4*, *6*
cvscoop-class, 4

deviance,scoopfit-method *(scoopfit-class)*, 10

fitted,scoopfit-method *(scoopfit-class)*, 10

genes *(basal)*, 2
group *(basal)*, 2
group.lasso *(scoop)*, 8

lasso *(scoop)*, 8

plot,cvscoop-method *(plot,cvscoop-methods)*, 5
plot,cvscoop-methods, 5
plot,scoopfit-method *(plot,scoopfit-methods)*, 6
plot,scoopfit-methods, 6
plot.cvscoop, *4*, *5*
plot.cvscoop *(plot,cvscoop-methods)*, 5
plot.scoop, *10*
plot.scoop *(plot,scoopfit-methods)*, 6
predict,scoopfit-method *(scoopfit-class)*, 10
print,scoopfit-method *(scoopfit-class)*, 10

residuals,scoopfit-method *(scoopfit-class)*, 10

scoop, *4*, *7*, 8, *11*, *13*
scoop-package, 2
scoopfit, *4*, *7*, *10*, *13*
scoopfit-class, 10
selection, *10*
selection *(selection,scoopfit-methods)*, 12
selection,scoopfit-method *(selection,scoopfit-methods)*, 12
selection,scoopfit-methods, 12
sparse.coop.lasso *(scoop)*, 8
sparse.group.lasso *(scoop)*, 8

tree.coop.lasso *(scoop)*, 8
tree.group.lasso *(scoop)*, 8

x *(basal)*, 2

`y` *(basal)*, 2