Package: sparseR (via r-universe)

October 4, 2026

Type Package

Title Variable Selection under Ranked Sparsity Principles for Interactions and Polynomials

Version 0.3.2

Description An implementation of ranked sparsity methods, including penalized regression methods such as the sparsity-ranked lasso, its non-convex alternatives, and elastic net, as well as the sparsity-ranked Bayesian Information Criterion. As described in Peterson and Cavanaugh (2022) <doi:10.1007/s10182-021-00431-7>, ranked sparsity is a philosophy with methods primarily useful for variable selection in the presence of prior informational asymmetry, which occurs in the context of trying to perform variable selection in the presence of interactions and/or polynomials. Ultimately, this package attempts to facilitate dealing with cumbersome interactions and polynomials while not avoiding them entirely. Typically, models selected under ranked sparsity principles will also be more transparent, having fewer falsely selected interactions and polynomials than other methods.

Suggests survival, knitr, rmarkdown, kableExtra, testthat, covr, modeldata, MASS

Imports ncvreg, rlang, magrittr, dplyr, recipes (>= 1.0.0)

Depends R (>= 3.5)

License GPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 7.3.2

URL https://petersonr.github.io/sparseR/, https://github.com/petersonR/sparseR/

Date 2025-04-14

Config/pak/sysreqs libicu-dev
Repository https://petersonr.r-universe.dev
Date/Publication 2025-04-14 17:57:15 UTC
RemoteUrl https://github.com/petersonr/sparser
RemoteRef HEAD
RemoteSha 225504235524f695586e4b26524dd04f2d2eec7b

Contents

sparseR-package . . . 2
datasets . . . 3
EBIC . . . 4
effect_plot . . . 5
get_penalties . . . 6
plot.sparseR . . . 7
predict.sparseR . . . 8
print.sparseR . . . 8
sparseR . . . 9
sparseR_prep . . . 11
sparseRBIC_bootstrap . . . 12
sparseRBIC_sampsplit . . . 13
sparseRBIC_step . . . 14
step_center_to . . . 16
summary.sparseR . . . 18

Index **19**

`sparseR-package` *sparseR: Implement ranked sparsity for selecting interactions and polynomials*

Description

The `sparseR` package implements various techniques for selecting from a set of interaction and polynomial terms under ranked sparsity. Additional tools for data pre-processing, post-selection inference, and visualization are also included.

Author(s)

Maintainer: Ryan Andrew Peterson <ryan.a.peterson@cuanschutz.edu> (ORCID)

See Also

Useful links:

- https://petersonr.github.io/sparseR/
- https://github.com/petersonR/sparseR/

datasets *Data sets*

Description

Detrano data sets (cleveland, hungarian, switzerland, va); The Iowa Radon Lung Cancer Study (irlcs_radon_syn): Data simulated to resemble the IRLCS study; Sheddon survival data (Z: clinical covariates, S:survival outcome)

Usage

```
cleveland

hungarian

switzerland

va

irlcs_radon_syn

Z

S
```

Format

An object of class `data.frame` with 303 rows and 14 columns.

An object of class `data.frame` with 294 rows and 14 columns.

An object of class `data.frame` with 123 rows and 14 columns.

An object of class `data.frame` with 200 rows and 14 columns.

An object of class `data.frame` with 1027 rows and 16 columns.

An object of class `data.frame` with 442 rows and 6 columns.

An object of class `Surv` with 442 rows and 2 columns.

Source

Detrano data https://archive.ics.uci.edu/ml/datasets/heart+disease

IRLCS data sets https://cheec.uiowa.edu/research/residential-radon-and-lung-cancer-case-control-stu

Sheddon https://www.gsea-msigdb.org/gsea/msigdb/cards/SHEDDEN_LUNG_CANCER_POOR_SURVIVAL_A6

References

Detrano Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., & Froelicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. American Journal of Cardiology, 64,304–310.

IRLCS FIELD, R., SMITH, B., STECK, D. et al. Residential radon exposure and lung cancer: Variation in risk estimates using alternative exposure scenarios. J Expo Sci Environ Epidemiol 12, 197–203 (2002). https://www.nature.com/articles/7500215

Shedden Director's Challenge Consortium for the Molecular Classification of Lung Adenocarcinoma, Shedden, K., Taylor, J. M., Enkemann, S. A., Tsao, M. S., Yeatman, T. J., Gerald, W. L., Eschrich, S., Jurisica, I., Giordano, T. J., Misek, D. E., Chang, A. C., Zhu, C. Q., Strumpf, D., Hanash, S., Shepherd, F. A., Ding, K., Seymour, L., Naoki, K., Pennell, N., ... Beer, D. G. (2008). Gene expression-based survival prediction in lung adenocarcinoma: a multi-site, blinded validation study. Nature medicine, 14(8), 822–827. https://www.nature.com/articles/nm.1790

`EBIC` *Custom IC functions for stepwise models*

Description

Custom IC functions for stepwise models

Usage

```
EBIC(...)

## Default S3 method:
EBIC(fit, varnames, pen_info, gammafn = NULL, return_df = TRUE, ...)

RBIC(fit, ...)

## Default S3 method:
RBIC(fit, varnames, pen_info, gammafn = NULL, return_df = TRUE, ...)

RAIC(fit, ...)

## Default S3 method:
RAIC(fit, varnames, pen_info, gammafn = NULL, return_df = TRUE, ...)
```

Arguments

`...`	additional args
`fit`	a fitted object
`varnames`	names of variables
`pen_info`	penalty information
`gammafn`	What to use for gamma in formula
`return_df`	should the deg. freedom be returned

Value

A vector of values for the criterion requested, and the degrees of freedom (appended to front of vector) if return_df == TRUE.

`effect_plot` *Plot relevant effects of a sparseR object*

Description

Plot relevant effects of a sparseR object

Usage

```
effect_plot(fit, ...)

## S3 method for class 'sparseR'
effect_plot(
  fit,
  coef_name,
  at = c("cvmin", "cv1se"),
  by = NULL,
  by_levels,
  nn = 101,
  plot.args = list(),
  resids = TRUE,
  legend_location = "bottomright",
  ...
)

## S3 method for class 'sparseRBIC'
effect_plot(
  fit,
  coef_name,
  by = NULL,
  by_levels,
  nn = 101,
  plot.args = list(),
  resids = TRUE,
  legend_location = "bottomright",
  ...
)
```

Arguments

`fit`	a ‘sparseR‘ object
`...`	additional arguments

coef_name	The name of the coefficient to plot along the x-axis
at	value of lambda to use
by	the variable(s) involved in the (possible) interaction
by_levels	values to cut continuous by variable (defaults to 3 quantiles)
nn	number of points to plot along prediction line
plot.args	list of arguments passed to the plot itself
resids	should residuals be plotted or not?
legend_location	location for legend passed to 'legend'

Value

nothing returned

Nothing (invisible) returned

`get_penalties` *Helper function to help set up penalties*

Description

Helper function to help set up penalties

Usage

```
get_penalties(
  varnames,
  poly,
  poly_prefix = "poly_",
  int_sep = "\\:",
  pool = FALSE,
  gamma = 0.5,
  cumulative_k = FALSE,
  cumulative_poly = TRUE
)
```

Arguments

varnames	names of the covariates in the model matrix
poly	max polynomial considered
poly_prefix	what comes before the polynomial specification in these varnames?
int_sep	What denotes the multiplication for interactions?
pool	Should polynomials and interactions be pooled?
gamma	How much should the penalty increase with group size (0.5 assumes equal contribution of prior information)

cumulative_k	Should penalties be increased cumulatively as order interaction increases? (only used if !pool)
cumulative_poly	Should penalties be increased cumulatively as order polynomial increases? (only used if !pool)

Details

This is primarily a helper function for sparseR, but it may be useful if doing the model matrix set up by hand.

Value

a list of relevant information for the variables, including:

penalties	the numeric value of the penalties
vartype	Variable type (main effect, order k interaction, etc)
varname	names of variables

---

plot.sparseR *Plot relevant properties of sparseR objects*

Description

Plot relevant properties of sparseR objects

Usage

```
## S3 method for class 'sparseR'
plot(x, plot_type = c("both", "cv", "path"), cols = NULL, log.l = TRUE, ...)
```

Arguments

x	a ‘sparseR‘ object
plot_type	should the solution path, CV results, or both be plotted?
cols	option to specify color of groups
log.l	should the x-axis (lambda) be logged?
...	extra plotting options

Value

nothing returned

predict.sparseR *Predict coefficients or responses for sparseR object*

Description

Predict coefficients or responses for sparseR object

Usage

```
## S3 method for class 'sparseR'
predict(object, newdata, lambda, at = c("cvmin", "cv1se"), ...)

## S3 method for class 'sparseR'
coef(object, lambda, at = c("cvmin", "cv1se"), ...)
```

Arguments

object	sparseR object
newdata	new data on which to make predictions
lambda	a particular value of lambda to predict with
at	a "smart" guess to use for lambda
...	additional arguments passed to predict.ncvreg

Value

predicted outcomes for ‘newdata‘ (or coefficients) at specified (or smart) lambda value

print.sparseR *Print sparseR object*

Description

Print sparseR object

Usage

```
## S3 method for class 'sparseR'
print(x, prep = FALSE, ...)
```

Arguments

x	a sparseR object
prep	Should the SR set-up information be printed as well?
...	additional arguments passed to print.ncvreg

Value

returns x invisibly

`sparseR` *Fit a ranked-sparsity model with regularized regression*

Description

Fit a ranked-sparsity model with regularized regression

Usage

```
sparseR(
  formula,
  data,
  family = c("gaussian", "binomial", "poisson", "coxph"),
  penalty = c("lasso", "MCP", "SCAD"),
  alpha = 1,
  ncvgamma = 3,
  lambda.min = 0.005,
  k = 1,
  poly = 2,
  gamma = 0.5,
  cumulative_k = FALSE,
  cumulative_poly = TRUE,
  pool = FALSE,
  ia_formula = NULL,
  pre_process = TRUE,
  model_matrix = NULL,
  y = NULL,
  poly_prefix = "_poly_",
  int_sep = "\\:",
  pre_proc_opts = c("knnImpute", "scale", "center", "otherbin", "none"),
  filter = c("nzv", "zv"),
  extra_opts = list(),
  ...
)
```

Arguments

`formula`	Names of the terms
`data`	Data
`family`	The family of the model
`penalty`	What penalty should be used (lasso, MCP, or SCAD)
`alpha`	The mix of L1 penalty (lower values introduce more L2 ridge penalty)

ncvgamma	The tuning parameter for ncvreg (for MCP or SCAD)
lambda.min	The minimum value to be used for lambda (as ratio of max, see ?ncvreg)
k	The maximum order of interactions to consider (default: 1; all pairwise)
poly	The maximum order of polynomials to consider (default: 2)
gamma	The degree of extremity of sparsity rankings (see details)
cumulative_k	Should penalties be increased cumulatively as order interaction increases?
cumulative_poly	Should penalties be increased cumulatively as order polynomial increases?
pool	Should interactions of order k and polynomials of order k+1 be pooled together for calculating the penalty?
ia_formula	formula to be passed to step_interact (for interactions, see details)
pre_process	Should the data be preprocessed (if FALSE, must provide model_matrix)
model_matrix	A data frame or matrix specifying the full model matrix (used if !pre_process)
y	A vector of responses (used if !pre_process)
poly_prefix	If model_matrix is specified, what is the prefix for polynomial terms?
int_sep	If model_matrix is specified, what is the separator for interaction terms?
pre_proc_opts	List of preprocessing steps (see details)
filter	The type of filter applied to main effects + interactions
extra_opts	A list of options for all preprocess steps (see details)
...	Additional arguments (passed to fitting function)

Details

Selecting gamma: higher values of gamma will penalize "group" size more. By default, this is set to 0.5, which yields equal contribution of prior information across orders of interactions/polynomials (this is a good default for most settings).

Additionally, setting cumulative_poly or cumulative_k to TRUE increases the penalty cumulatively based on the order of either polynomial or interaction.

The options that can be passed to pre_proc_opts are: - knnImpute (should missing data be imputed?) - scale (should data be standardized)? - center (should data be centered to the mean or another value?) - otherbin (should factors with low prevalence be combined?) - none (should no preprocessing be done? can also specify a null object)

The options that can be passed to extra_opts are:

- centers (named numeric vector which denotes where each covariate should be centered)
- center_fn (alternatively, a function can be specified to calculate center such as min or median)
- freq_cut, unique_cut (see ?step_nzv; these get used by the filtering steps)
- neighbors (the number of neighbors for knnImpute)
- one_hot (see ?step_dummy), this defaults to cell-means coding which can be done in regularized regression (change at your own risk)
- raw (should polynomials not be orthogonal? defaults to true because variables are centered and scaled already by this point by default)

ia_formula will by default interact all variables with each other up to order k. If specified, ia_formula will be passed as the terms argument to recipes::step_interact, so the help documentation for that function can be investigated for further assistance in specifying specific interactions.

Value

an object of class sparseR containing the following:

fit	the fit object returned by ncvreg
srprep	a recipes object used to prep the data
pen_factors	the factor multiple on penalties for ranked sparsity
results	all coefficients and penalty factors at minimum CV lambda
results_summary	a tibble of summary results at minimum CV lambda
results1se	all coefficients and penalty factors at lambda_1se
results1se_summary	a tibble of summary results at lambda_1se
data	the (unprocessed) data
family	the family argument (for non-normal, eg. poisson)
info	a list containing meta-info about the procedure

References

For fitting functionality, the ncvreg package is used; see Breheny, P. and Huang, J. (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. Ann. Appl. Statist., 5: 232-253.

---

sparseR_prep *Preprocess & create a model matrix with interactions + polynomials*

Description

Preprocess & create a model matrix with interactions + polynomials

Usage

```
sparseR_prep(
  formula,
  data,
  k = 1,
  poly = 1,
  pre_proc_opts = c("knnImpute", "scale", "center", "otherbin", "none"),
  ia_formula = NULL,
  filter = c("nzv", "zv"),
  extra_opts = list(),
  family = "gaussian"
)
```

Arguments

formula	A formula of the main effects + outcome of the model
data	A required data frame or tibble containing the variables in formula
k	Maximum order of interactions to *numeric* variables
poly	the maximum order of polynomials to consider
pre_proc_opts	A character vector specifying methods for preprocessing (see details)
ia_formula	formula to be passed to step_interact (for interactions, see details)
filter	which methods should be used to filter out variables with (near) zero variance? (see details)
extra_opts	extra options to be used for preprocessing
family	family passed from sparseR

Details

The pre_proc_opts acts as a wrapper for the corresponding procedures in the recipes package. The currently supported options that can be passed to pre_proc_opts are: knnImpute: Should k-nearest-neighbors be performed (if necessary?) scale: Should variables be scaled prior to creating interactions (does not scale factor variables or dummy variables) center: Should variables be centered (will not center factor variables or dummy variables) otherbin:

ia_formula will by default interact all variables with each other up to order k. If specified, ia_formula will be passed as the terms argument to recipes::step_interact, so the help documentation for that function can be investigated for further assistance in specifying specific interactions.

The methods specified in filter are important; filtering is necessary to cut down on extraneous polynomials and interactions (in cases where they really don't make sense). This is true, for instance, when using dummy variables in polynomials , or when using interactions of dummy variables that relate to the same categorical variable.

Value

an object of class recipe; see recipes::recipe()

---

sparseRBIC_bootstrap *Bootstrap procedure for stepwise regression*

Description

Runs bootstrap on models selection procedure using RBIC to find bootstrapped standard error (smoothed, see Efron 2014) as well as selection percentage across candidate variables. (experimental)

Usage

```
sparseRBIC_bootstrap(srbic_fit, B = 100, quiet = FALSE)
```

Arguments

srbic_fit	An object fitted by sparseRBIC_step
B	Number of bootstrap samples
quiet	Should the display of a progress bar be silenced?

Value

a list containing:

results	a tibble containing coefficients, p-values, selection pct
bootstraps	a tibble of bootstrapped coefficients

---

sparseRBIC_sampsplit *Sample split procedure for stepwise regression*

Description

Runs multiple on models selection procedures using RBIC to achieve valid inferential results post-selection

Usage

```
sparseRBIC_sampsplit(srbic_fit, S = 100, quiet = FALSE)
```

Arguments

srbic_fit	An object fitted by sparseRBIC_step
S	Number of splitting iterations
quiet	Should the display of a progress bar be silenced?

Value

a list containing:

results	a tibble containing coefficients, p-values, selection pct
splits	a tibble of different split-based coefficients

`sparseRBIC_step` *Fit a ranked-sparsity model with forward stepwise RBIC (experimental)*

Description

Fit a ranked-sparsity model with forward stepwise RBIC (experimental)

Usage

```
sparseRBIC_step(
  formula,
  data,
  family = c("gaussian", "binomial", "poisson"),
  k = 1,
  poly = 1,
  ic = c("RBIC", "RAIC", "BIC", "AIC", "EBIC"),
  hier = c("strong", "weak", "none"),
  sequential = (hier[1] != "none"),
  cumulative_k = FALSE,
  cumulative_poly = TRUE,
  pool = FALSE,
  ia_formula = NULL,
  pre_process = TRUE,
  model_matrix = NULL,
  y = NULL,
  poly_prefix = "_poly_",
  int_sep = "\\:",
  pre_proc_opts = c("knnImpute", "scale", "center", "otherbin", "none"),
  filter = c("nzv", "zv"),
  extra_opts = list(),
  trace = 0,
  message = TRUE,
  ...
)
```

Arguments

`formula`	Names of the terms
`data`	Data
`family`	The family of the model
`k`	The maximum order of interactions to consider
`poly`	The maximum order of polynomials to consider
`ic`	The information criterion to use
`hier`	Should hierarchy be enforced (weak or strong)? Must be set with sequential == TRUE (see details)

sequential	Should the main effects be considered first, orders sequentially added/considered?
cumulative_k	Should penalties be increased cumulatively as order interaction increases?
cumulative_poly	Should penalties be increased cumulatively as order polynomial increases?
pool	Should interactions of order k and polynomials of order k+1 be pooled together for calculating the penalty?
ia_formula	formula to be passed to step_interact via terms argument
pre_process	Should the data be preprocessed (if FALSE, must provide model_matrix)
model_matrix	A data frame or matrix specifying the full model matrix (used if !pre_process)
y	A vector of responses (used if !pre_process)
poly_prefix	If model_matrix is specified, what is the prefix for polynomial terms?
int_sep	If model_matrix is specified, what is the separator for interaction terms?
pre_proc_opts	List of preprocessing steps (see details)
filter	The type of filter applied to main effects + interactions
extra_opts	A list of options for all preprocess steps (see details)
trace	Should intermediate results of model selection process be output
message	should experimental message be suppressed
...	additional arguments for running stepwise selection

Details

This function mirrors sparseR but uses stepwise selection guided by RBIC.

Additionally, setting cumulative_poly or cumulative_k to TRUE increases the penalty cumulatively based on the order of either polynomial or interaction.

The hier hierarchy enforcement will only work if sequential == TRUE, and notably will only consider the "first gen" hierarchy, that is, that all main effects which make up an interaction are already in the model. It is therefore possible for a third order interaction (x1:x2:x3) to enter a model without x1:x2 or x2:x3, so long as x1, x2, and x3 are all in the model.

The options that can be passed to pre_proc_opts are:

- knnImpute (should missing data be imputed?)
- scale (should data be standardized)?
- center (should data be centered to the mean or another value?)
- otherbin (should factors with low prevalence be combined?)
- none (should no preprocessing be done? can also specify a null object)

The options that can be passed to extra_opts are:

- centers (named numeric vector which denotes where each covariate should be centered)
- center_fn (alternatively, a function can be specified to calculate center such as min or median)
- freq_cut, unique_cut (see ?step_nzv - these get used by the filtering steps)
- neighbors (the number of neighbors for knnImpute)

- one_hot (see ?step_dummy), this defaults to cell-means coding which can be done in regularized regression (change at your own risk)
- raw (should polynomials not be orthogonal? defaults to true because variables are centered and scaled already by this point by default)

Value

an object of class `sparseRBIC` containing the following:

`fit`	the final fit object
`srprep`	a `recipes` object used to prep the data
`pen_info`	coefficient-level variable counts, types + names
`data`	the (unprocessed) data
`family`	the family argument (for non-normal, eg. poisson)
`info`	a list containing meta-info about the procedure
`stats`	the IC for each fit and respective terms included

`step_center_to` *Centering numeric data to a value besides their mean*

Description

'step_center_to' generalizes 'step_center' to allow for a different function than the 'mean' function to calculate centers. It creates a *specification* of a recipe step that will normalize numeric data to have a 'center' of zero.

Usage

```
step_center_to(
  recipe,
  ...,
  role = NA,
  trained = FALSE,
  centers = NULL,
  center_fn = mean,
  na_rm = TRUE,
  skip = FALSE,
  id = rand_id("center_to")
)

## S3 method for class 'step_center_to'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See [selections()] for more details. For the ‘tidy‘ method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
centers	A named numeric vector of centers. This is ‘NULL‘ until computed by [prep.recipe()] (or it can be specified as a named numeric vector as well?).
center_fn	a function to be used to calculate where the center should be
na_rm	A logical value indicating whether ‘NA‘ values should be removed during computations.
skip	A logical. Should the step be skipped when the recipe is baked by [bake.recipe()]? While all operations are baked when [prep.recipe()] is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using ‘skip = TRUE‘ as it may affect the computations for subsequent operations
id	A character string that is unique to this step to identify it.
x	A ‘step_center_to‘ object.

Details

Centering data means that the average of a variable is subtracted from the data. ‘step_center_to‘ estimates the variable centers from the data used in the ‘training‘ argument of ‘prep.recipe‘. ‘bake.recipe‘ then applies the centering to new data sets using these centers.

Value

An updated version of ‘recipe‘ with the new step added to the sequence of existing steps (if any). For the ‘tidy‘ method, a tibble with columns ‘terms‘ (the selectors or variables selected) and ‘value‘ (the centers).

See Also

[recipe()] [prep.recipe()] [bake.recipe()]

Examples

```
data(biomass, package = "modeldata")

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipes::recipe(
 HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
 data = biomass_tr)
```

```
center_trans <- rec %>%
  step_center_to(carbon, contains("gen"), -hydrogen)

center_obj <- recipes::prep(center_trans, training = biomass_tr)

transformed_te <- recipes::bake(center_obj, biomass_te)

biomass_te[1:10, names(transformed_te)]
transformed_te

recipes::tidy(center_trans)
recipes::tidy(center_obj)
```

`summary.sparseR` *Summary of sparseR model coefficients*

Description

Summary of sparseR model coefficients

Usage

```
## S3 method for class 'sparseR'
summary(object, lambda, at = c("cvmin", "cv1se"), ...)
```

Arguments

`object`	a sparseR object
`lambda`	a particular value of lambda to predict with
`at`	a "smart" guess to use for lambda
`...`	additional arguments to be passed to summary.ncvreg

Value

an object of class ‘summary.ncvreg‘ at specified or smart value of lambda.

Index

∗ **datagen**
    step_center_to, 16
∗ **datasets**
    datasets, 3
∗ **normalization_methods**
    step_center_to, 16
∗ **preprocessing**
    step_center_to, 16

cleveland *(*datasets*)*, 3
coef.sparseR *(*predict.sparseR*)*, 8

datasets, 3

EBIC, 4
effect_plot, 5

get_penalties, 6

hungarian *(*datasets*)*, 3

irlcs_radon_syn *(*datasets*)*, 3

plot.sparseR, 7
predict.sparseR, 8
print.sparseR, 8

RAIC *(*EBIC*)*, 4
RBIC *(*EBIC*)*, 4
recipes::recipe(), *12*

S *(*datasets*)*, 3
sparseR, 9
sparseR-package, 2
sparseR_prep, 11
sparseRBIC_bootstrap, 12
sparseRBIC_sampsplit, 13
sparseRBIC_step, 14
step_center_to, 16
summary.sparseR, 18
switzerland *(*datasets*)*, 3

tidy.step_center_to *(*step_center_to*)*, 16

va *(*datasets*)*, 3

Z *(*datasets*)*, 3