

# Package ‘abess’

April 21, 2021

**Type** Package

**Title** Adaptive Best Subset Selection in Polynomial Time

**Version** 0.1.0

**Date** 2021-04-21

**Maintainer** Jin Zhu <zhu37@mail2.sysu.edu.cn>

**Description** Extremely efficient toolkit for solving the best subset selection problem in linear regression, logistic regression, Poisson regression, Cox proportional hazard model, multiple-response Gaussian, and multinomial regression. It implements and generalizes algorithms designed in <doi:10.1073/pnas.2014241117> that exploits a novel sequencing-and-splicing technique to guarantee exact support recovery and globally optimal solution in polynomial times.

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 2.10)

**Imports** Rcpp, MASS, methods, Matrix

**LinkingTo** Rcpp, RcppEigen

**RoxygenNote** 7.1.1

**SystemRequirements** C++11

**Suggests** testthat, knitr, rmarkdown

**VignetteBuilder** knitr

**URL** <https://github.com/abess-team/abess>,  
<https://abess-team.github.io/abess/>

**BugReports** <https://github.com/abess-team/abess/issues>

**NeedsCompilation** yes

**Author** Jin Zhu [aut, cre] (<<https://orcid.org/0000-0001-8550-5822>>),  
Kangkang Jiang [aut],  
Yanhang Zhang [aut],  
Liyuan Hu [aut],  
Junxian Zhu [aut],

Canhong Wen [aut],  
 Heping Zhang [aut] (<<https://orcid.org/0000-0002-0688-4076>>),  
 Xueqin Wang [aut] (<<https://orcid.org/0000-0001-5205-9950>>)

**Repository** CRAN

**Date/Publication** 2021-04-21 12:00:05 UTC

## R topics documented:

abess.default . . . . .	2
coef.abess . . . . .	7
deviance.abess . . . . .	8
extract . . . . .	9
generate.data . . . . .	10
plot.abess . . . . .	12
predict.abess . . . . .	13
print.abess . . . . .	14
trim32 . . . . .	15
<b>Index</b>	<b>16</b>

---

abess.default	<i>Adaptive Best-Subset Selection via Splicing</i>
---------------	--

---

## Description

Adaptive best-subset selection for regression, classification, counting-response, censored-response, multi-response modeling in polynomial times.

## Usage

```
## Default S3 method:
abess(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "cox", "mgaussian", "multinomial"),
  tune.path = c("sequence", "gsection"),
  tune.type = c("gic", "ebic", "bic", "aic", "cv"),
  weight = rep(1, nrow(x)),
  normalize = NULL,
  c.max = 2,
  support.size = NULL,
  gs.range = NULL,
  always.include = NULL,
  max.splicing.iter = 20,
  screening.num = NULL,
  warm.start = TRUE,
```

```

    n folds = 5,
    cov.update = TRUE,
    newton = c("exact", "approx"),
    newton.thresh = 1e-06,
    max.newton.iter = NULL,
    early.stop = FALSE,
    num.threads = 0,
    seed = 1,
    ...
)

## S3 method for class 'formula'
abess(formula, data, subset, na.action, ...)

```

### Arguments

x	Input matrix, of dimension $n \times p$ ; each row is an observation vector and each column is a predictor/feature/variable.
y	The response variable, of n observations. For family = "binomial" should have two levels. For family = "poisson", y should be a vector with positive integer. For family = "cox", y should be a two-column matrix with columns named time and status. For family = "mgaussian", y should be a matrix of quantitative responses. For family = "multinomial", y should be a factor of at least three levels. Note that, for either "binomial" or "multinomial", if y is presented as a numerical vector, it will be coerced into a factor.
family	One of the following models: "gaussian" (continuous response), "binomial" (binary response), "poisson" (non-negative count), "cox" (left-censored response), "mgaussian" (multivariate continuous response). Depending on the response. Any unambiguous substring can be given.
tune.path	The method to be used to select the optimal support size. For method = "sequence", we solve the best subset selection problem for each size in support.size. For method = "gsection", we solve the best subset selection problem with support size ranged in gs.range, where the specific support size to be considered is determined by golden section.
tune.type	The type of criterion for choosing the support size. Available options are "gic", "ebic", "bic", "aic" and "cv". Default is "gic".
weight	Observation weights. Default is 1 for each observation.
normalize	Options for normalization. normalize = 0 for no normalization. normalize = 1 for subtracting the mean of columns of x. normalize = 2 for scaling the columns of x to have $\sqrt{n}$ norm. normalize = 3 for subtracting the means of the columns of x and y, and also normalizing the columns of x to have $\sqrt{n}$ norm. If normalize = NULL, normalize will be set 1 for "gaussian", 2 for "binomial". Default is normalize = NULL.
c.max	an integer splicing size. Default is: c.max = 2.
support.size	An integer vector representing the alternative support sizes. Only used for method = "sequence". Default is $0:\min(n, \text{round}(n/(\log(\log(n))\log(p))))$ .

<code>gs.range</code>	A integer vector with two elements. The first element is the minimum model size considered by golden-section, the later one is the maximum one. Default is <code>gs.range = c(1, min(n, round(n/(log(log(n))log(p)))))</code> . Not available now.
<code>always.include</code>	An integer vector containing the indexes of variables that should always be included in the model.
<code>max.splicing.iter</code>	The maximum number of performing splicing algorithm. In most of the case, only a few times of splicing iteration can guarantee the convergence. Default is <code>max.splicing.iter = 20</code> .
<code>screening.num</code>	An integer number. Preserve <code>screening.num</code> number of predictors with the largest marginal maximum likelihood estimator before running algorithm.
<code>warm.start</code>	Whether to use the last solution as a warm start. Default is <code>warm.start = TRUE</code> .
<code>nfolds</code>	The number of folds in cross-validation. Default is <code>nfolds = 5</code> . For variables in the same group, they should be located in adjacent columns of <code>x</code> and their corresponding index in <code>group.index</code> should be the same. Denote the first group as 1, the second 2, etc. If you do not fit a model with a group structure, please set <code>group.index = NULL</code> . Default is <code>NULL</code> .
<code>cov.update</code>	A logical value only used for <code>family = "gaussian"</code> . If <code>cov.update = TRUE</code> , use a covariance-based implementation; otherwise, a naive implementation. The naive method is more efficient than covariance-based method only when $p \gg n$ . Default: <code>cov.update = TRUE</code> .
<code>newton</code>	A character specify the Newton's method for fitting generalized linear models, it should be either <code>newton = "exact"</code> or <code>newton = "approx"</code> . If <code>newton = "exact"</code> , then the exact hessian is used, while <code>newton = "approx"</code> uses diagonal entry of the hessian, and can be faster (especially when <code>family = "cox"</code> ).
<code>newton.thresh</code>	a numeric value for controlling positive convergence tolerance. The Newton's iterations converge when $ dev - dev_{old} /( dev  + 0.1) < newton.thresh$ .
<code>max.newton.iter</code>	a integer giving the maximal number of Newton's iteration iterations. Default is <code>max.newton.iter = 10</code> if <code>newton = "exact"</code> , and <code>max.newton.iter = 60</code> if <code>newton = "approx"</code> .
<code>early.stop</code>	A boolean value decide whether early stopping. If <code>early.stop = TRUE</code> , algorithm will stop if the last tuning value less than the existing one. Default: <code>early.stop = FALSE</code> .
<code>num.threads</code>	An integer decide the number of threads to be concurrently used for cross-validation (i.e., <code>tune.type = "cv"</code> ). If <code>num.threads = 0</code> , then all of available cores will be used. Default: <code>num.threads = 0</code> .
<code>seed</code>	Seed to be used to divide the sample into cross-validation folds. Default is <code>seed = 1</code> .
<code>...</code>	further arguments to be passed to or from methods.
<code>formula</code>	an object of class "formula": a symbolic description of the model to be fitted. The details of model specification are given in the "Details" section of "formula".

data	a data frame containing the variables in the formula.
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to <code>getOption("na.action")</code> .

### Details

Best-subset selection aims to find a small subset of predictors, so that the resulting model is expected to have the most desirable prediction accuracy. Best-subset selection problem under the support size  $s$  is

$$\min_{\beta} -2 \log L(\beta) \text{ s.t. } \|\beta\|_0 \leq s,$$

where  $L(\beta)$  is arbitrary convex functions. In the GLM case,  $\log L(\beta)$  is the log-likelihood function; in the Cox model,  $\log L(\beta)$  is the log partial-likelihood function.

The best subset selection problem is solved by the "abess" algorithm in this package, see Zhu (2020) for details. Under mild conditions, the algorithm exactly solve this problem in polynomial time. This algorithm exploits the idea of sequencing and splicing to reach a stable solution in finite steps when  $s$  is fixed. To find the optimal support size  $s$ , we provide various criterion like GIC, AIC, BIC and cross-validation error to determine it.

### Value

A S3 abess class object, which is a list with the following components:

beta	A $p$ -by- <code>length(support.size)</code> matrix of coefficients for univariate family, stored in column format; while a list of <code>length(support.size)</code> coefficients matrix (with size $p$ -by- <code>ncol(y)</code> ) for multivariate family.
intercept	An intercept vector of length <code>length(support.size)</code> for univariate family; while a list of <code>length(support.size)</code> intercept vector (with size <code>ncol(y)</code> ) for multivariate family.
dev	the deviance of length <code>length(support.size)</code> .
tune.value	A value of tuning criterion of length <code>length(support.size)</code> .
nobs	The number of sample used for training.
nvars	The number of variables used for training.
family	Type of the model.
tune.path	The path type for tuning parameters.
support.size	The actual <code>support.size</code> values used. Note that it is not necessary the same as the input if the later have non-integer values or duplicated values.
best.size	The best support size selected by the tuning value.
tune.type	The criterion type for tuning parameters.
tune.path	The strategy for tuning parameters.
screening.vars	The character vector specify the feature selected by feature screening. It would a empty character vector if <code>screening.num = 0</code> .
call	The original call to <code>abess</code> .

**Author(s)**

Jin Zhu, Junxian Zhu, Canhong Wen, Heping Zhang, Xueqin Wang

**References**

A polynomial algorithm for best-subset selection problem. Junxian Zhu, Canhong Wen, Jin Zhu, Heping Zhang, Xueqin Wang. Proceedings of the National Academy of Sciences Dec 2020, 117 (52) 33117-33123; DOI: 10.1073/pnas.2014241117

Sure independence screening for ultrahigh dimensional feature space. Fan, J. and Lv, J. (2008), Journal of the Royal Statistical Society: Series B (Statistical Methodology), 70: 849-911. <https://doi.org/10.1111/j.1467-9868.2008.00674.x>

Targeted Inference Involving High-Dimensional Data Using Nuisance Penalized Regression. Qiang Sun & Heping Zhang (2020). Journal of the American Statistical Association, DOI: 10.1080/01621459.2020.1737079

**See Also**

[print.abess](#), [predict.abess](#), [coef.abess](#), [extract.abess](#), [plot.abess](#), [deviance.abess](#).

**Examples**

```
n <- 100
p <- 20
support.size <- 3

##### linear model #####
dataset <- generate.data(n, p, support.size)
abess_fit <- abess(dataset[["x"]], dataset[["y"]])
## helpful generic functions:
print(abess_fit)
coef(abess_fit, support.size = 3)
predict(abess_fit, newx = dataset[["x"]][1:10, ],
        support.size = c(3, 4))
str(extract(abess_fit, 3))
deviance(abess_fit)
plot(abess_fit)

##### logistic model #####
dataset <- generate.data(n, p, support.size, family = "binomial")
## allow cross-validation to tuning
abess_fit <- abess(dataset[["x"]], dataset[["y"]],
                  family = "binomial", tune.type = "cv")
abess_fit

##### poisson model #####
dataset <- generate.data(n, p, support.size, family = "poisson")
abess_fit <- abess(dataset[["x"]], dataset[["y"]],
                  family = "poisson", tune.type = "cv")
abess_fit
```

```
##### Cox model #####
dataset <- generate.data(n, p, support.size, family = "cox")
abess_fit <- abess(dataset[["x"]], dataset[["y"]],
                  family = "cox", tune.type = "cv")

##### Multivariate gaussian model #####
dataset <- generate.data(n, p, support.size, family = "mgaussian")
abess_fit <- abess(dataset[["x"]], dataset[["y"]],
                  family = "mgaussian", tune.type = "cv")
plot(abess_fit, type = "l2norm")

##### Multinomial model (multi-classification) #####
dataset <- generate.data(n, p, support.size, family = "multinomial")
abess_fit <- abess(dataset[["x"]], dataset[["y"]],
                  family = "multinomial", tune.type = "cv")
predict(abess_fit, newx = dataset[["x"]][1:10, ],
        support.size = c(3, 4), type = "response")

##### Feature screening #####
p <- 1000
dataset <- generate.data(n, p, support.size)
abess_fit <- abess(dataset[["x"]], dataset[["y"]],
                  screening.num = 100)
str(extract(abess_fit))

##### Formula interface #####
data("trim32")
abess_fit <- abess(y ~ ., data = trim32)
abess_fit
```

---

coef.abess

*Extract Model Coefficients from a fitted "abess" object.*


---

## Description

This function provides estimated coefficients from a fitted "abess" object.

## Usage

```
## S3 method for class 'abess'
coef(object, support.size = NULL, sparse = TRUE, ...)
```

## Arguments

object	An "abess" project.
support.size	An integer vector specifies the coefficient fitted at given support.size. If support.size = NULL, then all coefficients would be returned. Default: support.size = NULL.

sparse            A logical value, specifying whether the coefficients should be presented as sparse matrix or not. Default: sparse = TRUE.

...                Other arguments.

**Value**

A coefficient matrix when fitting an univariate model including gaussian, binomial, poisson, and cox; otherwise, a list containing coefficient matrices. For a coefficient matrix, each row is a variable, and each column is a support size.

**See Also**

[print.abess](#), [predict.abess](#), [coef.abess](#), [extract.abess](#), [plot.abess](#), [deviance.abess](#).

---

deviance.abess	<i>Extract the deviance from a fitted "abess" object.</i>
----------------	---

---

**Description**

Similar to other deviance methods, which returns deviance from a fitted "abess" object.

**Usage**

```
## S3 method for class 'abess'
deviance(object, type = c("standard", "gic", "ebic", "bic", "aic"), ...)
```

**Arguments**

object            A "abess" object.

type              The type of deviance. One of the following: "standard", "gic", "ebic", "bic" and "aic". Default is "standard".

...                additional arguments

**Value**

A numeric vector.

**See Also**

[print.abess](#), [predict.abess](#), [coef.abess](#), [extract.abess](#), [plot.abess](#), [deviance.abess](#).



---

extract	<i>Extract one model from a fitted "abess" object.</i>
---------	--

---

**Description**

Extract the fixed-support-size model's information such as the selected predictors, coefficient estimation, and so on.

**Usage**

```
extract(object, support.size = NULL, ...)
```

```
## S3 method for class 'abess'
extract(object, support.size = NULL, ...)
```

**Arguments**

object	An "abess" project.
support.size	An integer value specifies the model size fitted at given support.size. If support.size = NULL, then the model with the best tuning value would be returned. Default: support.size = NULL.
...	Other arguments.

**Value**

A list object including the following components:

beta	A $p$ -by-1 matrix of sparse matrix, stored in column format.
intercept	The fitted intercept value.
support.size	The support.size used in the function.
support.beta	The support.size-length vector of fitted coefficients on the support set.
support.vars	The character vector gives variables in the support set.
tune.value	The tuning value of the model.
dev	The deviance of the model.

**See Also**

[print.abess](#), [predict.abess](#), [coef.abess](#), [extract.abess](#), [plot.abess](#), [deviance.abess](#).

---

generate.data	<i>Generate simulated data</i>
---------------	--------------------------------

---

## Description

Generate simulated data under the generalized linear model and Cox proportional hazard model.

## Usage

```
generate.data(
  n,
  p,
  support.size = NULL,
  rho = 0,
  family = c("gaussian", "binomial", "poisson", "cox", "mgaussian", "multinomial"),
  beta = NULL,
  cortype = 1,
  snr = 10,
  weibull.shape = 1,
  uniform.max = 1,
  y.dim = 3,
  class.num = 3,
  seed = 1
)
```

## Arguments

n	The number of observations.
p	The number of predictors of interest.
support.size	The number of nonzero coefficients in the underlying regression model. Can be omitted if beta is supplied.
rho	A parameter used to characterize the pairwise correlation in predictors. Default is 0.
family	The distribution of the simulated response. "gaussian" for univariate quantitative response, "binomial" for binary classification response, "poisson" for counting response, "cox" for left-censored response, "mgaussian" for multivariate quantitative response, "mgaussian" for multi-classification response.
beta	The coefficient values in the underlying regression model. If it is supplied, support.size would be omitted.
cortype	The correlation structure. cortype = 1 denotes the independence structure, where the covariance matrix has $(i, j)$ entry equals $I(i \neq j)$ . cortype = 2 denotes the exponential structure, where the covariance matrix has $(i, j)$ entry equals $\rho^{ i-j }$ . cortype = 3 denotes the constant structure, where the non-diagonal entries of covariance matrix are rho and diagonal entries are 1.

snr	A numerical value controlling the signal-to-noise ratio (SNR). The SNR is defined as the variance of $x\beta$ divided by the variance of a gaussian noise: $\frac{Var(x\beta)}{\sigma^2}$ . The gaussian noise $\epsilon$ is set with mean 0 and variance. The noise is added to the linear predictor $\eta = x\beta$ . Default is snr = 10.
weibull.shape	The shape parameter of the Weibull distribution. It works only when family = "cox". Default: weibull.shape = 1.
uniform.max	A parameter controlling censored rate. A large value implies a small censored rate; otherwise, a large censored rate. It works only when family = "cox". Default is uniform.max = 1.
y.dim	Response's Dimension. It works only when family = "mgaussian". Default: y.dim = 3.
class.num	The number of class. It works only when family = "multinomial". Default: class.num = 3.
seed	random seed. Default: seed = 1.

### Details

For family = "gaussian" , the data model is

$$Y = X\beta + \epsilon.$$

The underlying regression coefficient  $\beta$  has uniform distribution [m, 100m] and  $m = 5\sqrt{2\log(p)/n}$ .

For family= "binomial", the data model is

$$Prob(Y = 1) = \exp(X\beta + \epsilon)/(1 + \exp(X\beta + \epsilon)).$$

The underlying regression coefficient  $\beta$  has uniform distribution [2m, 10m] and  $m = 5\sqrt{2\log(p)/n}$ .

For family = "poisson", the data is modeled to have an exponential distribution:

$$Y = Exp(\exp(X\beta + \epsilon)).$$

The underlying regression coefficient  $\beta$  has uniform distribution [2m, 10m] and  $m = \sqrt{2\log(p)/n}/3$ .

For family = "cox", the model for failure time  $T$  is

$$T = (-\log(U/\exp(X\beta)))^{1/weibull.shape},$$

where  $U$  is a uniform random variable with range [0, 1]. The centering time  $C$  is generated from uniform distribution [0, uniform.max], then we define the censor status as  $\delta = I(T \leq C)$  and observed time as  $R = \min\{T, C\}$ . The underlying regression coefficient  $\beta$  has uniform distribution [2m, 10m], where  $m = 5\sqrt{2\log(p)/n}$ .

In the above models,  $\epsilon \sim N(0, \sigma^2)$ , where  $\sigma^2$  is determined by the snr.

### Value

A list object comprising:

x	Design matrix of predictors.
y	Response variable.
beta	The coefficients used in the underlying regression model.

**Author(s)**

Jin Zhu

**Examples**

```
# Generate simulated data
n <- 200
p <- 20
support.size <- 5
dataset <- generate.data(n, p, support.size)
str(dataset)
```

---

plot.abess

*Creat plot from a fitted "abess" object*


---

**Description**

Produces a coefficient/deviance/tuning-value plot for a fitted "abess" object.

**Usage**

```
## S3 method for class 'abess'
plot(
  x,
  type = c("coef", "l2norm", "dev", "dev.ratio", "tune"),
  label = FALSE,
  ...
)
```

**Arguments**

x	A "abess" object.
type	The type of terms to be plot in the y-axis. One of the following: "coef" (i.e., coefficients), "l2norm" (i.e., L2-norm of coefficients), "dev" (i.e., deviance), and "tune" (i.e., tuning value). Default is "coef".
label	A logical value. If label = TRUE (the default), label the curves with variable sequence numbers.
...	Other graphical parameters to plot

**Value**

No return value, called for side effects.

**Note**

If family = "mgaussian" or family = "multinomial", a coefficient plot is produced for each dimension of multivariate response.

**See Also**

[print.abess](#), [predict.abess](#), [coef.abess](#), [extract.abess](#), [plot.abess](#), [deviance.abess](#).

**Examples**

```
dataset <- generate.data(100, 20, 3)
abess_fit <- abess(dataset[["x"]], dataset[["y"]])
plot(abess_fit)
plot(abess_fit, type = "l2norm")
plot(abess_fit, type = "dev")
plot(abess_fit, type = "tune")
```

---

predict.abess	<i>Make predictions from a fitted "abess" object. Returns predictions from a fitted "abess" object.</i>
---------------	---

---

**Description**

Make predictions from a fitted "abess" object.

Returns predictions from a fitted "abess" object.

**Usage**

```
## S3 method for class 'abess'
predict(object, newx, type = c("link", "response"), support.size = NULL, ...)
```

**Arguments**

object	An "abess" project.
newx	New data used for prediction. If omitted, the fitted linear predictors are used.
type	type = "link" gives the linear predictors for "binomial", "poisson" or "cox" models; for "gaussian" models it gives the fitted values. type = "response" gives the fitted probabilities for "binomial", fitted mean for "poisson" and the fitted relative-risk for "cox"; for "gaussian", type = "response" is equivalent to type = "link"
support.size	An integer value specifies the model size fitted at given support.size. If support.size = NULL, then the model with the best tuning value would be returned. Default: support.size = NULL.
...	Additional arguments affecting the predictions produced.

**Value**

The object returned depends on the types of family.

**See Also**

[print.abess](#), [predict.abess](#), [coef.abess](#), [extract.abess](#), [plot.abess](#), [deviance.abess](#).

---

print.abess

*Print method for a fitted "abess" object*

---

**Description**

Prints the fitted model and returns it invisibly.

**Usage**

```
## S3 method for class 'abess'  
print(x, digits = max(5, getOption("digits") - 5), ...)
```

**Arguments**

x	A "abess" object.
digits	Minimum number of significant digits to be used.
...	additional print arguments

**Details**

Print a data.frame with three columns: the first column is support size of model; the second column is deviance of model; the last column is the tuning value of the certain tuning type.

**Value**

No return value, called for side effects

**See Also**

[print.abess](#), [predict.abess](#), [coef.abess](#), [extract.abess](#), [plot.abess](#), [deviance.abess](#).

---

trim32

*The Bardet-Biedl syndrome Gene expression data*

---

**Description**

Gene expression data (500 gene probes for 120 samples) from the microarray experiments of mammalian eye tissue samples of Scheetz et al. (2006).

**Format**

A data frame with 120 rows and 501 variables, where the first variable is the expression level of TRIM32 gene, and the remaining 500 variables are 500 gene probes.

**Details**

In this study, laboratory rats (*Rattus norvegicus*) were studied to learn about gene expression and regulation in the mammalian eye. Inbred rat strains were crossed and tissue extracted from the eyes of 120 animals from the F2 generation. Microarrays were used to measure levels of RNA expression in the isolated eye tissues of each subject. Of the 31,000 different probes, 18,976 were detected at a sufficient level to be considered expressed in the mammalian eye. For the purposes of this analysis, we treat one of those genes, Trim32, as the outcome. Trim32 is known to be linked with a genetic disorder called Bardet-Biedl Syndrome (BBS): the mutation (P130S) in Trim32 gives rise to BBS.

**Note**

This data set contains 120 samples with 500 predictors. The 500 predictors are features with maximum marginal correlation to Trim32 gene.

**References**

T. Scheetz, k. Kim, R. Swiderski, A. Philp, T. Braun, K. Knudtson, A. Dorrance, G. DiBona, J. Huang, T. Casavant, V. Sheffield, E. Stone. Regulation of gene expression in the mammalian eye and its relevance to eye disease. Proceedings of the National Academy of Sciences of the United States of America, 2006.

# Index

`abess` (`abess.default`), [2](#)  
`abess.default`, [2](#)

`coef.abess`, [6](#), [7](#), [8](#), [9](#), [13](#), [14](#)

`deviance.abess`, [6](#), [8](#), [8](#), [9](#), [13](#), [14](#)

`extract`, [9](#)  
`extract.abess`, [6](#), [8](#), [9](#), [13](#), [14](#)

`formula`, [4](#)

`generate.data`, [10](#)

`plot.abess`, [6](#), [8](#), [9](#), [12](#), [13](#), [14](#)  
`predict.abess`, [6](#), [8](#), [9](#), [13](#), [13](#), [14](#)  
`print.abess`, [6](#), [8](#), [9](#), [13](#), [14](#), [14](#)

`trim32`, [15](#)