Package 'DCEM'

April 6, 2019

Type Package

Title Clustering for Multivariate and Univariate Data Using Expectation Maximization Algorithm

Version 0.0.2

Maintainer Sharma Parichit <parishar@iu.edu>

Description Implements the Expectation Maximisation (EM) algorithm for clustering finite gaussian mixture models for
both multivariate and univariate datasets. The initialization is done by randomly selecting the samples from the
dataset as the mean of the Gaussian(s). This version improves the parameter initialization on big datasets.
The algorithm returns a set of Gaussian parameters-posterior probabilities, mean, covariance matrices
(multivariate data)/standard-deviation (for univariate datasets) and priors.
Reference: Hasan Kurban, Mark Jenne, Mehmet M. Dalkilic (2016) <doi:10.1007/s41060-017-0062-1>.
This work is partially supported by NCI Grant 1R01CA213466-01.

License GPL-3

Encoding UTF-8

LazyData true

Imports mvtnorm (>= 1.0.7), matrixcalc (>= 1.0.3), MASS (>= 7.3.49)

RoxygenNote 6.1.1

Depends R(>= 3.5.0)

URL https://github.iu.edu/parishar/DCEM

BugReports https://github.iu.edu/parishar/DCEM/issues

NeedsCompilation no

Author Sharma Parichit [aut, cre, ctb],
Kurban Hasan [aut, ctb],
Jenne Mark [aut, ctb],
Dalkilic Mehmet [aut]

Repository CRAN

Date/Publication 2019-04-05 23:02:42 UTC

R topics documented:

`cov_mv` *cov_mv: Part of DCEM package.*

Description

Initializes the co-variance matrices as the identity matrices for the Gaussian(s). The list will simply contain one co-variance matrix per Gaussian. If the user specifies 3 cluster(s) then there will be 3 entries in the list.

Usage

```
cov_mv(num_cov, numcol)
```

Arguments

`num_cov`	(numeric): Number of covariance matrices corresponding to the cluster(s).
`numcol`	(numeric): The number of columns in the dataset.

Details

The dimensions of each matrix will be numcol * numcol where numcol is the number of columns in the dataset.

Value

A list of identity matrices. The number of entries in the list is equal to the input parameter (num_cov).

The elements of the list can be accessed as - list[[1]] or list[[2]].

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work is partially supported by NCI Grant 1R01CA213466-01.

Examples

```
# Genrating the Identity matrix as the co-variance matrix.

cov_mv(2, 3)
cov_mv(10, 100)
```

DCEM *DCEM: Data clustering through Expectation-Maximization algorithm.*

Description

Implements Expectation-Maximization (EM) algorithm for clustering the univariate and multivariate finite Gaussian mixture data. Currently, the missing data is not imputed by the package and the user is expected to either remove features with missing values or impute them before using DCEM.

DCEM supports two initialisation schemes

1. **Random Initialisation:** Initializes the mean randomly. Refer means_uv and means_mv for initialisation on univariate and multivariate data respectively.
2. **Improved Initialisation:** Based on the Kmeans++ idea published in, K-means++: The Advantages of Careful Seeding, David Arthur and Sergei Vassilvitskii. URL http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf. See means_uv_impr and means_mv_impr for details.
3. This can be specified as the **seeding** parameter during the training. See dcem_train for further details.

Demonstration and Testing

Cleaning the data: The data should be cleaned (redundant columns should be removed). For example columns containing the labels or redundant entries (such as a column of only 0's or 1's). See trim_data for details on cleaning the data. Refer: dcem_test for more details.

Understanding the output of dcem_test

The function dcem_test() returns a list of objects. This list contains the parameters associated with the Gaussian(s), posterior probabilities (prob), mean (mean), co-variance (cov)/standard-deviation(sd) and priors.

Note: The routine dcem_test() is only for demonstration purpose. The function dcem_test calls the main routine dcem_train. See dcem_train for further details.

Accessing the output parameters

1 Posterior Probabilities: **sample_out$prob** (a matrix of posterior-probabilities for the points in the dataset.)

2 Mean(s): **sample_out$mean**

For multivariate data: It is a matrix of means for the Gaussians. Each row in the matrix corresponds to a mean for the Gaussian.

For univariate data: It is a vector of means. Each element of the vector corresponds to one Gaussian.

3 Co-variance matrices (in case of multivariate data): **sample_out$cov** (list of co-variance matrices for the Gaussians)

Standard-deviation (in case of univariate data): **sample_out$sd** (vector of standard deviation for the Gaussians)

4 Priors: **sample_out$prior** (a vector of priors for the Gaussians.)

How to run on your dataset

See `dcem_train` for examples.

Package organization

The package is organized as a set of preprocessing functions and the core clustering modules. These functions are briefly described below.

1. `trim_data`: This is used to remove the columns from the dataset. The user should clean the dataset before calling the dcem_train routine. **User can also clean the dataset themselves (without using trim_data) and then pass it to the dcem_train function**
2. `dcem_train`: This is the primary interface to the EM routine. It accepts the cleaned dataset and other parameters (number of iterations, convergence threshold etc.) and run the algorithm until:
   (a) The number of iterations is crossed.
   (b) The convergence threshold is achieved.

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work is partially supported by NCI Grant 1R01CA213466-01.

External Packages: DCEM requires R packages 'mvtnorm'[1] and 'matrixcalc'[2] for multivariate density calculation and for checking the matrix singularity respectively.

[1] Alan Genz, Frank Bretz, Tetsuhisa Miwa, Xuefei Mi, Friedrich Leisch, Fabian Scheipl, Torsten Hothorn (2019). mvtnorm: Multivariate Normal and t Distributions. R package version 1.0-7. URL http://CRAN.R-project.org/package=mvtnorm

[2] https://CRAN.R-project.org/package=matrixcalc

For improving the initialisation of cluster mean(s), the original idea published in [3] is used.

[3] k-means++: The Advantages of Careful Seeding, David Arthur and Sergei Vassilvitskii. URL http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

References

Using data to build a better EM: EM* for big data.

Hasan Kurban, Mark Jenne, Mehmet M. Dalkilic (2016) <doi:https://doi.org/10.1007/s41060-017-0062-1>.

`dcem_cluster_mv` *dcem_cluster (multivariate data): Part of DCEM package.*

Description

Implements the Expectation Maximization algorithm for multivariate data. This function is internally called by the dcem_train routine.

Usage

```
dcem_cluster_mv(data, mean_mat, cov_list, prior_vec, num, iteration_count,
threshold, numrows, numcols)
```

Arguments

`data`	A matrix: The dataset provided by the user.
`mean_mat`	(matrix): The matrix containing the initial mean(s) for the Gaussian(s).
`cov_list`	(list): A list containing the initial covariance matrices for the Gaussian(s).
`prior_vec`	(vector): A vector containing the initial priors for the Gaussian(s).
`num`	(numeric): The number of clusters specified by the user. Default value is 2.
`iteration_count`	(numeric): The number of iterations for which the algorithm should run, if the convergence is not achieved within the specified threshold then the algorithm stops and exits. Default: 200.
`threshold`	(numeric): A small value to check for convergence (if the estimated mean(s) are within this specified threshold then the algorithm stops and exit). **Note: Choosing a very small value (0.0000001) for threshold can increase the runtime substantially and the algorithm may not converge. On the other hand, choosing a larger value (0.1) can lead to sub-optimal clustering. Default: 0.00001**.
`numrows`	(numeric): Number of rows in the dataset (After processing the missing values).
`numcols`	(numeric): Number of columns in the dataset (After processing the missing values).

Value

A list of objects. This list contains parameters associated with the Gaussian(s) (posterior probabilities, mean, co-variance and priors)

1. Posterior Probabilities: **sample_out$prob** (a matrix of posterior-probabilities for the points in the dataset.)
2. Mean(s): **sample_out$mean**
   For multivariate data: It is a matrix of means for the Gaussian(s). Each row in the matrix corresponds to a mean for the Gaussian.
3. Co-variance matrices (in case of multivariate data): **sample_out$cov** (list of co-variance matrices for the Gaussian(s))
4. Priors: **sample_out$prior** (a vector of priors for the Gaussian(s).)

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work is partially supported by NCI Grant 1R01CA213466-01.

References

Using data to build a better EM: EM* for big data.

Hasan Kurban, Mark Jenne, Mehmet M. Dalkilic (2016) <doi:https://doi.org/10.1007/s41060-017-0062-1>.

`dcem_cluster_uv` *dcem_cluster_uv (univariate data): Part of DCEM package.*

Description

Implements the Expectation Maximization algorithm for the univariate data. This function is internally called by the dcem_train routine.

Usage

```
dcem_cluster_uv(data, mean_vector, sd_vector, prior_vec, num, iteration_count,
threshold, numrows, numcols)
```

Arguments

`data`	(matrix): The dataset provided by the user (converted to matrix format).
`mean_vector`	(vector): The vector containing the initial means of the Gaussians.
`sd_vector`	(vector): The vector containing the initial standard deviation for the Gaussians. The initial sd are set to be 1. They are updated during the iterations of the algorithm.

`prior_vec` (vector): The vector containing the initial priors for the Gaussians. They are initialized uniformly.

`num` (numeric): The number of clusters specified by the user. Default value is 2.

`iteration_count`
(numeric): The number of iterations for which the algorithm should run. if the convergence is not achieved within the specified threshold then the algorithm stops and exits. Default: 200.

`threshold` (numeric): A small value to check for convergence (if the estimated mean(s) are within this specified threshold then the algorithm stops and exit).

Note: Choosing a very small value (0.0000001) for threshold can increase the runtime substantially and the algorithm may not converge. On the other hand, choosing a larger value (0.1) can lead to sub-optimal clustering. Default: 0.00001.

`numrows` (numeric): Number of rows in the dataset (After processing the missing values).

`numcols` (numeric): Number of columns in the dataset (After processing the missing values).

Value

A list of objects. This list contains parameters associated with the Gaussian(s) (posterior probabilities, mean, co-variance/standard-deviation and priors)

1. Posterior Probabilities: **sample_out$prob** A matrix of posterior-probabilities
2. Mean(s): **sample_out$mean**
   For univariate data: It is a vector of means. Each element of the vector corresponds to one Gaussian.
3. Standard-deviation(s): **sample_out$sd**
   For univariate data: Vector of standard deviation for the Gaussian(s))
4. Priors: **sample_out$prior** A vector of priors for the Gaussian(s).

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work is partially supported by NCI Grant 1R01CA213466-01.

References

Hasan Kurban, Mark Jenne, Mehmet M. Dalkilic (2016) <doi:https://doi.org/10.1007/s41060-017-0062-1>.

dcem_test *dcem_test: Part of DCEM package.*

Description

For demonstrating the execution on the bundled dataset.

Usage

```
dcem_test()
```

Details

The dcem_test performs the following steps in order:

1. Read the data from the disk (from the file data/ionosphere_data.csv). The data folder is under the package installation folder.
2. The dataset details can be see by typing `ionosphere_data` in R-console or at https://archive.ics.uci.edu/ml/datasets/ionosphere.
3. Clean the data (by removing the columns). **The data should be cleaned before use.** Refer `trim_data` to see what columns should be removed and how. The package provides the basic interface for removing columns.
4. Call the `dcem_train` on the cleaned data.

Accessing the output parameters

The function dcem_test() calls dcem_train() that returns a list of objects. This list contains parameters associated with the Gaussian (posterior probabilities, mean, covariance/standard-deviation and priors). The parameters can be accessed as follows where sample_out is the list containing the output:

1 Posterior Probabilities: **sample_out$prob** A matrix of posterior-probabilities

2 Mean(s): **sample_out$mean**

For multivariate data: It is a matrix of means for the Gaussian(s). Each row in the matrix corresponds to a mean for the Gaussian.

For univariate data: It is a vector of means. Each element of the vector corresponds to one Gaussian.

3 Co-variance matrices: **sample_out$cov**

For multivariate data: List of co-variance matrices for the Gaussian(s).

Standard-deviation: **sample_out$sd**

For univariate data: Vector of standard deviation for the Gaussian(s))

4 Priors: **sample_out$prior** A vector of priors for the Gaussian(s).

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work is partially supported by NCI Grant 1R01CA213466-01.

References

Using data to build a better EM: EM* for big data.

Hasan Kurban, Mark Jenne, Mehmet M. Dalkilic (2016) <doi:https://doi.org/10.1007/s41060-017-0062-1>.

`dcem_train` *dcem_train: Part of DCEM package.*

Description

Learn the Gaussian parameters, mean and co-variance from the dataset. It calls the relevant clustering routine internally `dcem_cluster_uv` (univariate data) and `dcem_cluster_mv` (multivariate data).

Usage

```
dcem_train(data, threshold, iteration_count,  num_clusters, seeding)
```

Arguments

`data`	(dataframe): The dataframe containing the data. See `trim_data` for cleaning the data.
`threshold`	(decimal): A value to check for convergence (if the means are within this value then the algorithm stops and exit). **Default: 0.00001**.
`iteration_count`	(numeric): The number of iterations for which the algorithm should run, if the convergence is not achieved within the specified count then the algorithm stops and exit. **Default: 200**.
`num_clusters`	(numeric): The number of clusters. Default: **2**
`seeding`	(string): The initialisation scheme ('rand', 'improved'). Default: **rand**

Value

A list of objects. This list contains parameters associated with the Gaussian(s) (posterior probabilities, mean, standard-deviation and priors). The parameters can be accessed as follows where sample_out is the list containing the output:

1 Posterior Probabilities: **sample_out$prob** A matrix of posterior-probabilities

2 Mean(s): **sample_out$mean**

For multivariate data: It is a matrix of means for the Gaussian(s). Each row in the matrix corresponds to a mean for the Gaussian.

For univariate data: It is a vector of means. Each element of the vector corresponds to one Gaussian.

3 Co-variance matrices: **sample_out$cov**

For multivariate data: List of co-variance matrices for the Gaussian(s).

Standard-deviation: **sample_out$sd**

For univariate data: Vector of standard deviation for the Gaussian(s))

4 Priors: **sample_out$prior** A vector of priors for the Gaussian(s).

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work is partially supported by NCI Grant 1R01CA213466-01.

References

Using data to build a better EM: EM* for big data.

Hasan Kurban, Mark Jenne, Mehmet M. Dalkilic (2016) <doi:https://doi.org/10.1007/s41060-017-0062-1>.

Examples

```
# Simulating a mixture of univariate samples from three distributions
# with mean as 20, 70 and 100 and standard deviation as 10, 100 and 40 respectively.
sample_uv_data = as.data.frame(c(rnorm(100, 20, 10), rnorm(70, 70, 100), rnorm(50, 100, 40)))

# Randomly shuffle the samples.
sample_uv_data = as.data.frame(sample_uv_data[sample(nrow(sample_uv_data)),])

# Calling the dcem_train() function on the simulated data with threshold of
# 0.000001, iteration count of 1000 and random seeding respectively.
sample_uv_out = dcem_train(sample_uv_data, num_clusters = 3, iteration_count = 100,
threshold = 0.001)

# Simulating a mixture of multivariate samples from 2 gaussian distributions.
sample_mv_data = as.data.frame(rbind(MASS::mvrnorm(n=100, rep(2,5), Sigma = diag(5)),
MASS::mvrnorm(n=50, rep(14,5), Sigma = diag(5))))

# Calling the dcem_train() function on the simulated data with threshold of
# 0.00001, iteration count of 100 and random seeding method respectively.
sample_mv_out = dcem_train(sample_mv_data, threshold = 0.001, iteration_count = 100)

sample_mv_out$mean
#[1,]  2.053163  2.023351  2.017288  1.999596  1.983142
#[2,] 13.948244 14.010651 13.897140 14.285898 13.752592
```

ionosphere_data *Ionosphere data: A dataset of 351 radar readings*

Description

This dataset contains 351 entries (radar readings from a system in goose bay laboratory) and 35 columns. The 35th columns is the label columns identifying the entry as either good or bad. Additionally, the 2nd column only contains 0's.

Usage

```
ionosphere_data
```

Format

A file with 351 rows and 35 columns of multivariate data in a csv file. All values are numeric.

Source

Space Physics Group Applied Physics Laboratory Johns Hopkins University Johns Hopkins Road Laurel, MD 20723 Web URL: https://archive.ics.uci.edu/ml/datasets/ionosphere

References: Sigillito, V. G., Wing, S. P., Hutton, L. V., & Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Technical Digest, 10, 262-266.

means_mv *means_mv: Part of DCEM package.*

Description

Initialize the mean(s) for the Gaussian(s) by randomly selecting the samples from the dataset. This is the **default** method for initialising the means(s).

Usage

```
# Randomly seeding the mean(s).
means_mv(data, num_means)
```

Arguments

data	(matrix): The dataset provided by the user (converted to matrix format).
num_means	(numeric): The number of means (meu).

Value

A matrix containing the selected samples from the dataset. The initial means will be updated during the iterations of the algorithm.

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work was partially supported by NCI Grant 1R01CA213466-01.

Examples

```
# Generate random samples from a multivariate distribution.
sample_data = MASS::mvrnorm(n=10, rep(10,5), Sigma = diag(5))

# Randomly selecting the mean(s) from the data.
means_mv(sample_data, num_means=2)
```

`means_mv_impr` *means_mv_impr: Part of DCEM package.*

Description

Initialize the mean(s) for the Gaussian(s) by randomly selecting the samples from the dataset. It uses the proposed implementation from K-means++: The Advantages of Careful Seeding, David Arthur and Sergei Vassilvitskii. URL http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf.

Usage

```
# Randomly seeding the mean(s).
means_mv_impr(data, num_means)
```

Arguments

`data`	(matrix): The dataset provided by the user (converted to matrix format).
`num_means`	(numeric): The number of means (meu).

Value

A matrix containing the selected samples from the dataset. The initial means will be updated during the iterations of the algorithm.

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work was partially supported by NCI Grant 1R01CA213466-01.

Examples

```
# Generate random samples from a multivariate distribution.
sample_data = MASS::mvrnorm(n=10, rep(10,5), Sigma = diag(5))

# Randomly selecting the mean(s) from the data.
means_mv_impr(sample_data, num_means=2)
```

---

means_uv *means_uv: Part of DCEM package.*

Description

This function is internally called by the dcem_train to initialize the mean(s) for the Gaussian(s). It randomly selects the mean(s) from the range min(data):max(data). This is the **default** method for initialising the means(s).

Usage

```
# Randomly seeding the mean(s).
means_uv(data, num_means)
```

Arguments

data	(matrix): The dataset provided by the user (converted to matrix format).
num_means	(number): The number of means (meu).

Value

A vector containing the selected samples from the dataset. The initial means will be updated during the execution.

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work is partially supported by NCI Grant 1R01CA213466-01.

Examples

```
# Randomly selecting the samples from a normal distribution as inital mean(s).
means_uv(rnorm(100,20,10), 2)
```

means_uv_impr *means_uv_impr: Part of DCEM package.*

Description

This function is internally called by the dcem_train to initialize the mean(s) for the Gaussian(s). It uses the proposed implementation from K-means++: The Advantages of Careful Seeding, David Arthur and Sergei Vassilvitskii. URL http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf.

Usage

```
# Seeding the means using the K-means++ implementation.
means_uv_impr(data, num_means)
```

Arguments

data	(matrix): The dataset provided by the user (converted to matrix format).
num_means	(number): The number of means (meu).

Value

A vector containing the selected samples from the dataset. The initial means will be updated during the execution.

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work is partially supported by NCI Grant 1R01CA213466-01.

Examples

```
# Selecting the inital mean(s) using the non-uniform distance weighting.
# k-means++: The Advantages of Careful Seeding, David Arthur
# and Sergei Vassilvitskii.
# URL http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf.
means_uv_impr(rnorm(100,20,10), 2)
```

priors *priors: Part of DCEM package.*

Description

Initializes the prior vectors for the Gaussian(s)

Usage

```
priors(num)
```

Arguments

num	(numeric): Number of priors.

Details

For example, if the user specify 2 priors then the vector will have 2 entries (one for each cluster) where each will be 1/2 or 0.5.

Value

A vector of uniformly initialized prior values (numeric).

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work was partially supported by NCI Grant 1R01CA213466-01.

Examples

```
#Randomly generating the priors.

priors(2)
priors(3)
```

sd_uv *sd_uv: Part of DCEM package.*

Description

Initializes the standard deviation for the Gaussian(s).

Usage

```
sd_uv(data, num_sd)
```

Arguments

data	(matrix): The dataset provided by the user (converted to matrix format).
num_sd	(number): Number of values corresponding to the number of clusters.

Value

A vector of standard deviation value(s).

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic

This work was partially supported by NCI Grant 1R01CA213466-01.

Examples

```
# Standard deviation of a random sample.

sd_uv(rnorm(100,20,10), 2)
```

trim_data *trim_data: Part of DCEM package.*

Description

Removes the specified column(s) from the dataset.

Usage

```
trim_data(columns, data)
```

Arguments

columns	(string): A comma separated list of column(s) that needs to be removed from the dataset. Default: ”
data	(dataframe): Dataframe containing the input data.

Value

A dataframe with the specified column(s) removed from it.

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic This work is partially supported by NCI Grant 1R01CA213466-01.

References

Using data to build a better EM: EM* for big data.

Hasan Kurban, Mark Jenne, Mehmet M. Dalkilic (2016) <doi:https://doi.org/10.1007/s41060-017-0062-1>.

Examples

```
# Remove a range of columns. Generally, the columns containing the labels or
# redundant values (such as all 0's) should be removed before training the model.

trim_data("1,2", data.frame(x1=sample(1:100,10),
x2=sample(500:1000, 10), x3=sample(-100:0,10)))

# Remove a single column.

trim_data("2", data.frame(x1=sample(1:100,10),
x2=sample(500:1000, 10), x3=sample(-100:0,10)))
```

---

validate_data *validate_data: Part of DCEM package.*

Description

Implements sanity check for the input data. This function is for internal use and is called by the dcem_train.

Usage

```
validate_data(columns, numcols)
```

Arguments

columns	(string): A comma separated list of columns that needs to be removed from the dataset. Default: ”
numcols	(numeric): Number of columns in the dataset.

Details

An example would be to check if the column to be removed exist or not? trim_data internally calls this function before removing the column(s).

Value

boolean: TRUE if the columns exists otherwise FALSE.

Author(s)

Parichit Sharma <parishar@iu.edu>, Hasan Kurban, Mark Jenne, Mehmet Dalkilic This work is partially supported by NCI Grant 1R01CA213466-01.

References

Using data to build a better EM: EM* for big data.

Hasan Kurban, Mark Jenne, Mehmet M. Dalkilic (2016) <doi:https://doi.org/10.1007/s41060-017-0062-1>.

Examples

```
#Generate a dataframe with 2 columns containing random values.

# Check a range of columns.
validate_data("2,3,4", ncol(data.frame(x1=sample(1:100,10),
x2=sample(500:1000, 10), x3=sample(-100:0,10))))

# Check a single column.

validate_data("2", ncol(data.frame(x1=sample(1:100,10),
x2=sample(500:1000, 10))))
```

Index