

# Package ‘RAFS’

July 10, 2023

**Title** Robust Aggregative Feature Selection

**Version** 0.2.3

**Date** 2023-07-08

**URL** <https://www.mdfs.it/>

**Description** A cross-validated minimal-optimal feature selection algorithm.

It utilises popularity counting, hierarchical clustering with feature dissimilarity measures, and prefiltering with all-relevant feature selection method to obtain the minimal-optimal set of features.

**Depends** R (>= 4.2.0)

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Imports** fastcluster, MDFS (>= 1.5.3), splitTools

**NeedsCompilation** no

**Author** Radosław Piliszek [aut, cre]

**Maintainer** Radosław Piliszek <radoslaw.piliszek@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-07-10 19:00:18 UTC

## R topics documented:

builtin_dist_funs . . . . .	2
compute_fs_results . . . . .	2
cor_dist . . . . .	3
create_seeded_folds . . . . .	4
default_dist_funs . . . . .	4
default_fs_fun . . . . .	5
default_hclust_methods . . . . .	5
get_rafs_all_reps_from_popcnts . . . . .	6
get_rafs_occurrence_matrix . . . . .	6
get_rafs_reps_popcnts . . . . .	7

get_rafs_rep_tuples_matrix . . . . .	8
get_rafs_rep_tuples_popcnts . . . . .	9
get_rafs_tops_popcnts . . . . .	10
get_rafs_top_reps_from_popcnts . . . . .	10
get_rafs_top_rep_tuples_from_popcnts . . . . .	11
get_run_id . . . . .	12
run_rafs . . . . .	12
run_rafs_with_fs_results . . . . .	13
stig_dist . . . . .	14
stig_from_ig_dist . . . . .	15
stig_stable_dist . . . . .	16
vi_dist . . . . .	16

<b>Index</b>	<b>18</b>
--------------	-----------

---

builtin_dist_funs	<i>All built-in feature dissimilarity functions</i>
-------------------	---

---

### Description

To be used in [run\\_rafs](#).

### Usage

`builtin_dist_funs`

### Format

An object of class `list` of length 5.

### Details

See also [default\\_dist\\_funs](#).

---

<code>compute_fs_results</code>	<i>Compute preliminary feature selection results for RAFS</i>
---------------------------------	---

---

### Description

This is a secondary function, useful when experimenting with different feature selection filters and rankings. Its output is used in [run\\_rafs\\_with\\_fs\\_results](#) and it is called for the user in [run\\_rafs](#).

### Usage

`compute_fs_results(data, decision, k, seeds, fs_fun = default_fs_fun)`

**Arguments**

data	input data where columns are variables and rows are observations (all numeric)
decision	decision variable as a binary sequence of length equal to number of observations
k	number of folds for internal cross validation
seeds	a vector of seeds used for fold generation for internal cross validation
fs_fun	function to compute feature selection p-values, it must have the same signature as <a href="#">default_fs_fun</a> (which is the default, see its help to learn more)

**Value**

A [list](#) with feature selection results, e.g. from [default\\_fs\\_fun](#).

**Examples**

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
fs_results <- compute_fs_results(madelon$data, madelon$decision, 2, c(12345))
run_rafs_with_fs_results(madelon$data, madelon$decision, fs_results)
```

---

cor\_dist

*Feature dissimilarity based on Pearson's Correlation (cor)*


---

**Description**

To be used as one of the `dist_funs` in [run\\_rafs](#).

**Usage**

```
cor_dist(relevant_train_data, train_decision = NULL, seed = NULL)
```

**Arguments**

relevant_train_data	input data where columns are variables and rows are observations (all numeric); assumed to contain only relevant data
train_decision	decision variable as a binary sequence of length equal to number of observations
seed	a numerical seed

**Value**

A matrix of distances (dissimilarities).

create\_seeded\_folds     *Create seeded folds*

---

**Description**

A utility function used in RAFS but useful also for external cross-validation.

**Usage**

```
create_seeded_folds(decision, k, seed)
```

**Arguments**

decision	decision variable as a binary sequence of length equal to number of observations
k	number of folds for cross validation
seed	a numerical seed

**Value**

A vector of folds. Each fold being a vector of selected indices.

---

default\_dist\_funs     *Default feature dissimilarity functions*

---

**Description**

As used in [run\\_rafs](#).

**Usage**

```
default_dist_funs
```

**Format**

An object of class `list` of length 3.

**Details**

The default functions compute: Pearson's correlation (cor: [cor\\_dist](#)), Variation of Information (vi: [vi\\_dist](#)) and Symmetric Target Information Gain (stig: [stig\\_dist](#)).

These functions follow a similar protocol to [default\\_fs\\_fun](#). They expect the same input except for the assumption that the data passed in is relevant. Each of them outputs a matrix of distances (dissimilarities) between features.

See also [builtin\\_dist\\_funs](#).

---

default_fs_fun	<i>Default (example) feature selection function for RAFS</i>
----------------	--

---

**Description**

See [run\\_rafs](#) for how it is used. Only the train portion of the dataset is to be fed into this function.

**Usage**

```
default_fs_fun(train_data, train_decision, seed)
```

**Arguments**

train_data	input data where columns are variables and rows are observations (all numeric)
train_decision	decision variable as a binary sequence of length equal to number of observations
seed	a numerical seed

**Details**

The function **MUST** use this `train_data` and **MAY** ignore the `train_decision`.

If the function depends on randomness, it **MUST** use the `seed` parameter to seed the PRNG.

The function needs to return a [list](#) with at least two elements: `rel_vars` and `rel_vars_rank`, which are vectors and contain, respectively, the indices of variables considered relevant and the rank for each relevant variable. The function **MAY** return a list with more elements.

Other examples of sensible functions are included in the tests of this package.

**Value**

A [list](#) with at least two fields: `rel_vars` and `rel_vars_rank`, which are vectors and contain, respectively, the indices of variables considered relevant and the rank for each relevant variable.

---

default_hclust_methods	<i>Default hclust methods</i>
------------------------	-------------------------------

---

**Description**

As used in [run\\_rafs](#) to call [hclust](#).

**Usage**

```
default_hclust_methods
```

**Format**

An object of class character of length 4.

---

```
get_rafs_all_reps_from_popcnts
```

*Get all representatives from their popcnts*

---

### Description

This helper function works on results of `get_rafs_reps_popcnts` to obtain all representatives at the chosen number of clusters.

### Usage

```
get_rafs_all_reps_from_popcnts(reps_popcnts, n_clusters)
```

### Arguments

<code>reps_popcnts</code>	representatives' popcnts for the chosen variant as obtained from <code>get_rafs_reps_popcnts</code>
<code>n_clusters</code>	the desired number of clusters

### Value

A vector of all representatives.

### Examples

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
rafs_results <- run_rafs(madelon$data, madelon$decision, 2, c(12345))
rafs_reps_popcnts <- get_rafs_reps_popcnts(rafs_results, 5)
get_rafs_all_reps_from_popcnts(rafs_reps_popcnts$stig_single, 5)
```

---

```
get_rafs_occurrence_matrix
```

*Get co-occurrence matrix from RAFS results*

---

### Description

This function obtains a matrix describing a graph of co-occurrence at each count of clusters (from `n_clusters_range`) computed over all runs of RAFS.

### Usage

```
get_rafs_occurrence_matrix(
  rafs_results,
  interesting_reps,
  n_clusters_range = 2:15
)
```

### Arguments

`rafs_results` RAFS results as obtained from `run_rafs`  
`interesting_reps` the interesting representatives to build matrices for (in principle, these need not be representatives but it is more common)  
`n_clusters_range` range of clusters number to obtain matrices for

### Details

If a single result over a cluster number range is desired, the selected matrices can be summed.

### Value

A nested `list` with matrices. The first level is per the RAFS variant (combination of feature dissimilarity function and `hclust` method). The second level is per the number of clusters. The third (and last) level is the co-occurrence matrix.

### Examples

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
rafs_results <- run_rafs(madelon$data, madelon$decision, 2, c(12345))
rafs_reps_popcnts <- get_rafs_reps_popcnts(rafs_results, 5)
rafs_top_reps <- get_rafs_top_reps_from_popcnts(rafs_reps_popcnts$stig_single, 5)
get_rafs_occurrence_matrix(rafs_results, rafs_top_reps, 5)
```

---

`get_rafs_reps_popcnts` *Get representatives' popularity counts (popcnts) from RAFS results*

---

### Description

This function obtains popularity counts (`popcnts`) of representatives present at each count of clusters (from `n_clusters_range`) computed over all runs of RAFS.

### Usage

```
get_rafs_reps_popcnts(rafs_results, n_clusters_range = 2:15)
```

### Arguments

`rafs_results` RAFS results as obtained from `run_rafs`  
`n_clusters_range` range of clusters number to obtain `popcnts` for

**Details**

These results might be fed into further helper functions: [get\\_rafs\\_top\\_reps\\_from\\_popcnts](#) and [get\\_rafs\\_all\\_reps\\_from\\_popcnts](#).

**Value**

A nested [list](#) with popcnts. The first level is per the RAFS variant (combination of feature dissimilarity function and hclust method). The second level is per the number of clusters. The third (and last) level is popcnts per representative.

**Examples**

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
rafs_results <- run_rafs(madelon$data, madelon$decision, 2, c(12345))
get_rafs_reps_popcnts(rafs_results, 2:5)
```

---

```
get_rafs_rep_tuples_matrix
```

*Get representatives' tuples' co-representation matrix from RAFS results*

---

**Description**

This function obtains a matrix of representatives's describing a graph of co-representation at each count of clusters (from `n_clusters_range`) computed over all runs of RAFS.

**Usage**

```
get_rafs_rep_tuples_matrix(
  rafs_results,
  interesting_reps,
  n_clusters_range = 2:15
)
```

**Arguments**

```
rafs_results    RAFS results as obtained from run\_rafs
interesting_reps
                 the interesting representatives to build matrices for
n_clusters_range
                 range of clusters number to obtain matrices for
```

**Details**

If a single result over a cluster number range is desired, the selected matrices can be summed.



**Value**

A nested `list` with matrices. The first level is per the RAFS variant (combination of feature dissimilarity function and `hclust` method). The second level is per the number of clusters. The third (and last) level is the co-representation matrix.

**Examples**

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
rafs_results <- run_rafs(madelon$data, madelon$decision, 2, c(12345))
rafs_reps_popcnts <- get_rafs_reps_popcnts(rafs_results, 5)
rafs_top_reps <- get_rafs_top_reps_from_popcnts(rafs_reps_popcnts$stig_single, 5)
get_rafs_rep_tuples_matrix(rafs_results, rafs_top_reps, 5)
```

---

```
get_rafs_rep_tuples_popcnts
```

*Get representatives' tuples' popularity counts (popcnts) from RAFS results*

---

**Description**

This function obtains popularity counts (popcnts) of representatives' tuples present at each count of clusters (from `n_clusters_range`) computed over all runs of RAFS.

**Usage**

```
get_rafs_rep_tuples_popcnts(rafs_results, n_clusters_range = 2:15)
```

**Arguments**

```
rafs_results    RAFS results as obtained from run_rafs
n_clusters_range
                range of clusters number to obtain popcnts for
```

**Value**

A nested `list` with popcnts. The first level is per the RAFS variant (combination of feature dissimilarity function and `hclust` method). The second level is per the number of clusters. The third (and last) level is popcnts per representatives' tuple.

**Examples**

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
rafs_results <- run_rafs(madelon$data, madelon$decision, 2, c(12345))
get_rafs_rep_tuples_popcnts(rafs_results, 2:5)
```

---

get\_rafs\_tops\_popcnts *Get top popularity counts (popcnts) from FS results*

---

### Description

This function obtains popularity counts (popcnts) of top variables computed over all runs of FS.

### Usage

```
get_rafs_tops_popcnts(fs_results, n_top_range = 2:15)
```

### Arguments

fs\_results      RAFS FS results as obtained from [compute\\_fs\\_results](#)  
n\_top\_range      range of top number to obtain popcnts for

### Details

These results might be fed into further helper functions: [get\\_rafs\\_top\\_reps\\_from\\_popcnts](#) and [get\\_rafs\\_all\\_reps\\_from\\_popcnts](#).

### Value

A nested [list](#) with popcnts. The first level is per the number of top variables. The second (and last) level is popcnts per top variable.

### Examples

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
fs_results <- compute_fs_results(madelon$data, madelon$decision, 2, c(12345))
get_rafs_tops_popcnts(fs_results, 2:5)
```

---

get\_rafs\_top\_reps\_from\_popcnts  
*Get top (i.e., most common) representatives from their popcnts*

---

### Description

This helper function works on results of [get\\_rafs\\_reps\\_popcnts](#) to obtain the desired number of top (most common) representatives at the chosen number of clusters.

### Usage

```
get_rafs_top_reps_from_popcnts(reps_popcnts, n_clusters, n_reps = n_clusters)
```

**Arguments**

reps\_popcnts    popcnts for the chosen variant as obtained from [get\\_rafs\\_reps\\_popcnts](#)  
 n\_clusters      the desired number of clusters  
 n\_reps          the desired number of top representatives

**Value**

A vector of top representatives.

**Examples**

```

library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
rafs_results <- run_rafs(madelon$data, madelon$decision, 2, c(12345))
rafs_reps_popcnts <- get_rafs_reps_popcnts(rafs_results, 5)
get_rafs_top_reps_from_popcnts(rafs_reps_popcnts$stig_single, 5)

```

---

```
get_rafs_top_rep_tuples_from_popcnts
```

*Get top (i.e., most common) representatives's tuples from their popcnts*

---

**Description**

This helper function works on results of [get\\_rafs\\_rep\\_tuples\\_popcnts](#) to obtain the desired number of top (most common) representatives' tuples at the chosen number of clusters.

**Usage**

```

get_rafs_top_rep_tuples_from_popcnts(
  rep_tuples_popcnts,
  n_clusters,
  n_tuples = 1
)

```

**Arguments**

rep\_tuples\_popcnts    tuples' popcnts for the chosen variant as obtained from [get\\_rafs\\_rep\\_tuples\\_popcnts](#)  
 n\_clusters            the desired number of clusters  
 n\_tuples              the desired number of top tuples

**Value**

A list of top tuples (each tuple being a vector of representatives).

**Examples**

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
rafs_results <- run_rafs(madelon$data, madelon$decision, 2, c(12345))
rafs_rep_tuples_popcnts <- get_rafs_rep_tuples_popcnts(rafs_results, 5)
get_rafs_top_rep_tuples_from_popcnts(rafs_rep_tuples_popcnts$stig_single, 5)
```

---

get_run_id	<i>Generate CV run identifiers</i>
------------	------------------------------------

---

**Description**

A utility function used in RAFS to generate cross validation run identifiers, thus useful also for external cross-validation.

**Usage**

```
get_run_id(seed, k, i)
```

**Arguments**

seed	a numerical seed
k	number of folds for cross validation
i	current fold number (1 to k)

**Value**

A string with the run identifier.

---

run_rafs	<i>Robust Aggregative Feature Selection (RAFS)</i>
----------	--

---

**Description**

This is the main function of the RAFS library to run for analysis.

**Usage**

```
run_rafs(
  data,
  decision,
  k = 5,
  seeds = sample.int(32767, 10),
  fs_fun = default_fs_fun,
  dist_funs = default_dist_funs,
  hclust_methods = default_hclust_methods
)
```

**Arguments**

data	input data where columns are variables and rows are observations (all numeric)
decision	decision variable as a binary sequence of length equal to number of observations
k	number of folds for internal cross validation
seeds	a vector of seeds used for fold generation for internal cross validation
fs_fun	function to compute feature selection p-values, it must have the same signature as <a href="#">default_fs_fun</a> (which is the default, see its help to learn more)
dist_funs	a list of feature dissimilarity functions computed over the relevant portion of the training dataset (see the example <a href="#">default_dist_funs</a> and <a href="#">builtin_dist_funs</a> to learn more)
hclust_methods	a vector of <a href="#">hclust</a> methods to use

**Details**

Depending on your pipeline, you may want to also check out [run\\_rafs\\_with\\_fs\\_results](#) and [compute\\_fs\\_results](#) which this function simply wraps over.

The results from this function can be fed into one of the helper functions to analyse them further: [get\\_rafs\\_reps\\_popcnts](#), [get\\_rafs\\_rep\\_tuples\\_popcnts](#), [get\\_rafs\\_rep\\_tuples\\_matrix](#) and [get\\_rafs\\_occurrence\\_matrix](#).

**Value**

A nested [list](#) with [hclust](#) results. The first level is per the cross validation run. The second level is per the feature dissimilarity function. The third (and last) level is per the hclust method.

**Examples**

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
run_rafs(madelon$data, madelon$decision, 2, c(12345))
```

---

```
run_rafs_with_fs_results
```

*Robust Aggregative Feature Selection (RAFS) from feature selection results*

---

**Description**

This is a secondary function, useful when experimenting with different feature selection filters and rankings. The output is exactly the same as from [run\\_rafs](#).

**Usage**

```
run_rafs_with_fs_results(
  data,
  decision,
  fs_results,
  dist_funs = default_dist_funs,
  hclust_methods = default_hclust_methods
)
```

**Arguments**

`data` input data where columns are variables and rows are observations (all numeric)

`decision` decision variable as a binary sequence of length equal to number of observations

`fs_results` output from `compute_fs_results` computed for the same data and decision

`dist_funs` a list of feature dissimilarity functions computed over the relevant portion of the training dataset (see the example `default_dist_funs` to learn more)

`hclust_methods` a vector of `hclust` methods to use

**Value**

A nested `list` with `hclust` results. The first level is per the cross validation run. The second level is per the feature dissimilarity function. The third (and last) level is per the `hclust` method.

**Examples**

```
library(MDFS)
mdfs_omp_set_num_threads(1) # only to pass CRAN checks
data(madelon)
fs_results <- compute_fs_results(madelon$data, madelon$decision, 2, c(12345))
run_rafs_with_fs_results(madelon$data, madelon$decision, fs_results)
```

---

stig\_dist

*Symmetric Target Information Gain (STIG) computed directly*

---

**Description**

To be used as one of the `dist_funs` in `run_rafs`.

**Usage**

```
stig_dist(relevant_train_data, train_decision, seed)
```

**Arguments**

relevant\_train\_data      input data where columns are variables and rows are observations (all numeric); assumed to contain only relevant data

train\_decision      decision variable as a binary sequence of length equal to number of observations

seed                  a numerical seed

**Details**

This function computes the STIG metric directly from the data, maximising it over 30 discretisations.

**Value**

A matrix of distances (dissimilarities).

---

stig_from_ig_dist	<i>Symmetric Target Information Gain (STIG) computed from single Information Gains (IGs)</i>
-------------------	--

---

**Description**

To be used as one of the dist\_funs in [run\\_rafs](#).

**Usage**

```
stig_from_ig_dist(relevant_train_data, train_decision, seed)
```

**Arguments**

relevant\_train\_data      input data where columns are variables and rows are observations (all numeric); assumed to contain only relevant data

train\_decision      decision variable as a binary sequence of length equal to number of observations

seed                  a numerical seed

**Details**

This function computes the STIG metric from single Information Gains (IGs) maximised over 30 discretisations and then summed pair-wise.

This function is similar to [stig\\_dist](#) but the results differ slightly. We recommend the direct computation in general.

**Value**

A matrix of distances (dissimilarities).

---

stig_stable_dist	<i>Symmetric Target Information Gain (STIG) computed directly but with pre-computed 1D conditional entropy (aka stable)</i>
------------------	---

---

**Description**

To be used as one of the dist\_funs in [run\\_rafs](#).

**Usage**

```
stig_stable_dist(relevant_train_data, train_decision, seed)
```

**Arguments**

relevant_train_data	input data where columns are variables and rows are observations (all numeric); assumed to contain only relevant data
train_decision	decision variable as a binary sequence of length equal to number of observations
seed	a numerical seed

**Details**

This function computes the STIG metric directly from the data, maximising it over 30 discretisations, but reusing the common 1D conditional entropy.

**Value**

A matrix of distances (dissimilarities).

---

vi_dist	<i>Variation of Information (VI)</i>
---------	--------------------------------------

---

**Description**

To be used as one of the dist\_funs in [run\\_rafs](#).

**Usage**

```
vi_dist(relevant_train_data, train_decision = NULL, seed)
```

**Arguments**

relevant_train_data	input data where columns are variables and rows are observations (all numeric); assumed to contain only relevant data
train_decision	decision variable as a binary sequence of length equal to number of observations
seed	a numerical seed



**Details**

This function computes the Variation of Information (VI) averaged over 30 discretisations.

**Value**

A matrix of distances (dissimilarities).

# Index

## \* datasets

- builtin\_dist\_funs, 2
- default\_dist\_funs, 4
- default\_hclust\_methods, 5

builtin\_dist\_funs, 2, 4, 13

compute\_fs\_results, 2, 10, 13, 14

cor\_dist, 3, 4

create\_seeded\_folds, 4

default\_dist\_funs, 2, 4, 13, 14

default\_fs\_fun, 3, 4, 5, 13

default\_hclust\_methods, 5

get\_rafs\_all\_reps\_from\_popcnts, 6, 8, 10

get\_rafs\_occurrence\_matrix, 6, 13

get\_rafs\_rep\_tuples\_matrix, 8, 13

get\_rafs\_rep\_tuples\_popcnts, 9, 11, 13

get\_rafs\_reps\_popcnts, 6, 7, 10, 11, 13

get\_rafs\_top\_rep\_tuples\_from\_popcnts,  
11

get\_rafs\_top\_reps\_from\_popcnts, 8, 10,  
10

get\_rafs\_tops\_popcnts, 10

get\_run\_id, 12

hclust, 5, 13, 14

list, 3, 5, 7–10, 13, 14

run\_rafs, 2–5, 7–9, 12, 13–16

run\_rafs\_with\_fs\_results, 2, 13, 13

stig\_dist, 4, 14, 15

stig\_from\_ig\_dist, 15

stig\_stable\_dist, 16

vi\_dist, 4, 16