# Package 'admix'

October 12, 2022

**Title** Package Admix for Admixture (aka Contamination) Models

**Version** 0.4.0

**Description** Implements several methods to estimate the unknown quantities related to two-component admixture models, where the two components can belong to any distribution (note that in the case of multinomial mixtures, the two components must belong to the same family). Estimation methods depend on the assumptions made on the unknown component density (see Bordes and Vandekerkhove (2010) <doi:10.3103/S1066530710010023>; Patra and Sen (2016) <doi:10.1111/rssb.12148>); Milhaud, Pommeret, Salhi and Vandekerkhove (2021) <doi:10.1016/j.jspi.2021.05.010>). In practice, one can estimate both the mixture weight and the unknown component density in a wide variety of frameworks. On top of that, hypothesis tests can be performed in one and two-samples contexts to test the unknown component density. Finally, clustering of unknown mixture components is also feasible in a K-samples setting.

**License** GPL (>= 3)

**URL** https://github.com/XavierMilhaud/admix

**BugReports** https://github.com/XavierMilhaud/admix/issues

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**Imports** base, fdrtool, graphics, Iso, latex2exp, MASS, methods, orthopolynom, pracma, Rcpp, stats, utils

**Suggests** rmutil, doParallel, foreach, evd, logitnorm, flexsurv, plyr, reshape2, gridExtra, lattice, testthat (>= 3.0.0), knitr, rmarkdown, markdown

**Depends** R (>= 2.10)

**LinkingTo** Rcpp

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Xavier Milhaud [aut, cre],
Pierre Vandekerkhove [ctb],
Denys Pommeret [ctb],
Yahia Salhi [ctb]

**Maintainer** Xavier Milhaud <xavier.milhaud.research@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-03-24 12:30:05 UTC

# R topics documented:

---

`admix_clustering` *Clustering of K populations following admixture models*

---

## Description

Create clusters on the unknown components related to the K populations following admixture models. Based on the K-sample test using Inversion - Best Matching (IBM) approach, see 'Details' below for further information.

## Usage

```
admix_clustering(
  samples = NULL,
  n_sim_tab = 100,
  comp.dist = NULL,
  comp.param = NULL,
  parallel = FALSE,
  n_cpu = 2
)
```

## Arguments

| | |
|---|---|
| `samples` | A list of the K observed samples to be clustered, all following admixture distributions. |
| `n_sim_tab` | Number of simulated gaussian processes used in the tabulation of the inner convergence distribution in the IBM approach. |
| `comp.dist` | A list with 2*K elements corresponding to the component distributions (specified with R native names for these distributions) involved in the K admixture models. Elements, grouped by 2, refer to the unknown and known components of each admixture model, If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows with K = 3: list(f1 = NULL, g1 = 'norm', f2 = NULL, g2 = 'norm', f3 = NULL, g3 = 'rnorm'). |

| | |
|---|---|
| comp.param | A list with 2*K elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Elements, grouped by 2, refer to the parameters of unknown and known components of each admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows (with K = 3): list(f1 = NULL, g1 = list(mean=0,sd=1), f2 = NULL, g2 = list(mean=3,sd=1.1), f3 = NULL, g3 = list(mean=-2,sd=0.6)). |
| parallel | (default to FALSE) Boolean to indicate whether parallel computations are performed (speed-up the tabulation). |
| n_cpu | (default to 2) Number of cores used when parallelizing. |

## Details

See the paper at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

A list with three elements: 1) the identified clusters; 2) the cluster affiliation; 3) the discrepancy matrix.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data (chosen parameters indicate 2 clusters (populations (1,3), (2,4))!):
list.comp <- list(f1 = "gamma", g1 = "exp",
                  f2 = "gamma", g2 = "exp",
                  f3 = "gamma", g3 = "gamma",
                  f4 = "gamma", g4 = "exp")
list.param <- list(f1 = list(shape = 16, rate = 4), g1 = list(rate = 1/3.5),
                   f2 = list(shape = 14, rate = 2), g2 = list(rate = 1/5),
                   f3 = list(shape = 16, rate = 4), g3 = list(shape = 12, rate = 2),
                   f4 = list(shape = 14, rate = 2), g4 = list(rate = 1/7))
A.sim <- rsimmix(n=2600, unknownComp_weight=0.8, comp.dist = list(list.comp$f1,list.comp$g1),
                 comp.param = list(list.param$f1, list.param$g1))$mixt.data
B.sim <- rsimmix(n=3000, unknownComp_weight=0.7, comp.dist = list(list.comp$f2,list.comp$g2),
                 comp.param = list(list.param$f2, list.param$g2))$mixt.data
C.sim <- rsimmix(n=3500, unknownComp_weight=0.6, comp.dist = list(list.comp$f3,list.comp$g3),
                 comp.param = list(list.param$f3, list.param$g3))$mixt.data
D.sim <- rsimmix(n=4800, unknownComp_weight=0.5, comp.dist = list(list.comp$f4,list.comp$g4),
                 comp.param = list(list.param$f4, list.param$g4))$mixt.data
## Look for the clusters:
list.comp <- list(f1 = NULL, g1 = "exp",
                  f2 = NULL, g2 = "exp",
                  f3 = NULL, g3 = "gamma",
                  f4 = NULL, g4 = "exp")
list.param <- list(f1 = NULL, g1 = list(rate = 1/3.5),
```

```
                    f2 = NULL, g2 = list(rate = 1/5),
                    f3 = NULL, g3 = list(shape = 12, rate = 2),
                    f4 = NULL, g4 = list(rate = 1/7))
clusters <- admix_clustering(samples = list(A.sim,B.sim,C.sim,D.sim), n_sim_tab = 8,
                      comp.dist=list.comp, comp.param=list.param, parallel=FALSE, n_cpu=2)
clusters$clustering
```

---

`admix_estim` *Estimate the unknown parameters of the admixture model(s) under study*

---

## Description

Estimate the component weights, the location shift parameter (in case of a symmetric unknown component density), and the unknown component distribution using different estimation techniques. We remind that the i-th admixture model has probability density function (pdf) l_i such that: l_i = p_i * f_i + (1-p_i) * g_i, where g_i is the known component density. The unknown quantities p_i and f_i then have to be estimated.

## Usage

```
admix_estim(
  samples = NULL,
  sym.f = FALSE,
  est.method = c("PS", "BVdk", "IBM"),
  comp.dist = NULL,
  comp.param = NULL
)
```

## Arguments

| | |
|---|---|
| `samples` | A list of the K samples to be studied, all following admixture distributions. |
| `sym.f` | A boolean indicating whether the unknown component densities are assumed to be symmetric or not. |
| `est.method` | The estimation method to be applied. Can be one of 'BVdk' (Bordes and Vandekerkhove estimator), 'PS' (Patra and Sen estimator), or 'IBM' (Inversion Best-Matching approach). The same estimation method is performed on each sample. Important note: estimation by 'IBM' is unbiased only under H0, meaning that choosing this method requires to perform previously the test hypothesis between the pairs of samples. For further details, see section 'Details' below. |
| `comp.dist` | A list with 2*K elements corresponding to the component distributions (specified with R native names for these distributions) involved in the K admixture models. Elements, grouped by 2, refer to the unknown and known components of each admixture model, If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows with |

K = 3: list(f1 = NULL, g1 = 'norm', f2 = NULL, g2 = 'norm', f3 = NULL, g3 = 'rnorm').

`comp.param` A list with 2*K elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Elements, grouped by 2, refer to the parameters of unknown and known components of each admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows (with K = 3): list(f1 = NULL, g1 = list(mean=0,sd=1), f2 = NULL, g2 = list(mean=3,sd=1.1), f3 = NULL, g3 = list(mean=-2,sd=0.6)).

## Details

For further details on the different estimation techniques, see i) IBM approach at https://hal.archives-ouvertes.fr/hal-03201760 ; ii) Patra and Sen estimator: Patra, R.K. and Sen, B. (2016); Estimation of a Two-component Mixture Model with Applications to Multiple Testing; JRSS Series B, 78, pp. 869–893. ; iii) BVdk estimator: Bordes, L. and Vandekerkhove, P. (2010); Semiparametric two-component mixture model when a component is known: an asymptotically normal estimator; Math. Meth. Stat.; 19, pp. 22–41.

## Value

A list containing the estimated weight of every unknown component distribution among admixture samples.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
##### On a simulated example to see whether the true parameters are well estimated.
list.comp <- list(f1 = "norm", g1 = "norm",
                  f2 = "norm", g2 = "norm")
list.param <- list(f1 = list(mean = 0, sd = 1), g1 = list(mean = 2, sd = 0.7),
                   f2 = list(mean = 0, sd = 1), g2 = list(mean = -3, sd = 1.1))
## Simulate data:
sim1 <- rsimmix(n = 2100, unknownComp_weight = 0.8, comp.dist = list(list.comp$f1,list.comp$g1),
                comp.param = list(list.param$f1, list.param$g1))$mixt.data
sim2 <- rsimmix(n= 2000, unknownComp_weight = 0.85, comp.dist = list(list.comp$f2,list.comp$g2),
                comp.param = list(list.param$f2, list.param$g2))$mixt.data
## Estimate the mixture weights of the admixture models:
list.comp <- list(f1 = NULL, g1 = "norm",
                  f2 = NULL, g2 = "norm")
list.param <- list(f1 = NULL, g1 = list(mean = 2, sd = 0.7),
                   f2 = NULL, g2 = list(mean = -3, sd = 1.1))
estim <- admix_estim(samples = list(sim1,sim2), sym.f = TRUE, est.method = 'IBM',
                     comp.dist = list.comp, comp.param = list.param)
estim
```

admix_test *Hypothesis test between unknown components of the admixture models under study*

## Description

Perform hypothesis test between unknown components of a list of admixture models, where we remind that the i-th admixture model has probability density function (pdf) l_i such that: l_i = p_i * f_i + (1-p_i) * g_i, with g_i the known component density. The unknown quantities p_i and f_i are thus estimated, leading to the test given by the following null and alternative hypothesis: H0: f_i = f_j for all i != j against H1 : there exists at least i != j such that f_i differs from f_j. The test can be performed using two methods, either the comparison of coefficients obtained through polynomial basis expansions of the component densities, or by the inner-convergence property obtained using the IBM approach. See 'Details' below for further information.

## Usage

```
admix_test(
  samples = NULL,
  sym.f = FALSE,
  test.method = c("Poly", "ICV"),
  sim_U = NULL,
  n_sim_tab = 50,
  min_size = NULL,
  comp.dist = NULL,
  comp.param = NULL,
  support = c("Real", "Integer", "Positive", "Bounded.continuous"),
  parallel = FALSE,
  n_cpu = 4
)
```

## Arguments

| | |
|---|---|
| samples | A list of the K samples to be studied, all following admixture distributions. |
| sym.f | A boolean indicating whether the unknown component densities are assumed to be symmetric or not. |
| test.method | The testing method to be applied. Can be either 'Poly' (polynomial basis expansion) or 'ICV' (inner convergence from IBM). The same testing method is performed between all samples. In the one-sample case, only 'Poly' is available and the test is a gaussianity test. For further details, see section 'Details' below. |
| sim_U | (Used only with 'ICV' testing method, otherwise useless) Random draws of the inner convergence part of the contrast as defined in the IBM approach (see 'Details' below). |
| n_sim_tab | (Used only with 'ICV' testing method, otherwise useless) Number of simulated gaussian processes used in the tabulation of the inner convergence distribution in the IBM approach. |

| | |
|---|---|
| min_size | (Potentially used with 'ICV' testing method, otherwise useless) Minimal size among all samples (needed to take into account the correction factor for the variance-covariance assessment). |
| comp.dist | A list with 2*K elements corresponding to the component distributions (specified with R native names for these distributions) involved in the K admixture models. Elements, grouped by 2, refer to the unknown and known components of each admixture model, If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows with K = 3: list(f1 = NULL, g1 = 'norm', f2 = NULL, g2 = 'norm', f3 = NULL, g3 = 'rnorm'). |
| comp.param | A list with 2*K elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Elements, grouped by 2, refer to the parameters of unknown and known components of each admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows (with K = 3): list(f1 = NULL, g1 = list(mean=0,sd=1), f2 = NULL, g2 = list(mean=3,sd=1.1), f3 = NULL, g3 = list(mean=-2,sd=0.6)). |
| support | (Potentially used with 'Poly' testing method, otherwise useless) The support of the observations; one of "Real", "Integer", "Positive", or "Bounded.continuous". |
| parallel | (default to FALSE) Boolean indicating whether parallel computations are performed (speed-up the tabulation). |
| n_cpu | (default to 2) Number of cores used when parallelizing. |

## Details

For further details on hypothesis techniques, see i) Inner convergence through IBM approach at https://hal.archives-ouvertes.fr/hal-03201760 ; ii) Polynomial expansions at 'False Discovery Rate model Gaussianity test' (EJS, Pommeret & Vanderkerkhove, 2017), or 'Semiparametric two-sample admixture components comparison test: the symmetric case' (JSPI, Milhaud & al., 2021).

## Value

A list containing...

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
##### On a simulated example, with 1 sample (gaussianity test):
list.comp <- list(f1 = "norm", g1 = "norm")
list.param <- list(f1 = list(mean = 0, sd = 1), g1 = list(mean = 2, sd = 0.7))
## Simulate data:
sim1 <- rsimmix(n = 300, unknownComp_weight = 0.85, comp.dist = list(list.comp$f1,list.comp$g1),
                comp.param = list(list.param$f1, list.param$g1))$mixt.data
## Perform the test hypothesis:
list.comp <- list(f1 = NULL, g1 = "norm")
```

```
list.param <- list(f1 = NULL, g1 = list(mean = 2, sd = 0.7))
admix_test(samples = list(sim1), sym.f = TRUE, test.method = 'Poly', sim_U = NULL, n_sim_tab=50,
         min_size = NULL, comp.dist = list.comp, comp.param = list.param, support = "Real",
           parallel = FALSE, n_cpu = 2)
```

---

`allGalaxies` *Four galaxies (Carina, Sextans, Sculptor, Fornax) measurements of heliocentric velocities from SIMBAD astronomical database*

---

## Description

Four galaxies (Carina, Sextans, Sculptor, Fornax) measurements of heliocentric velocities from SIMBAD astronomical database

## Usage

```
allGalaxies
```

## Format

An evolving data frame of velocities for 4 dSph galaxies; namely Carina, Sextans, Sculptor and Fornax. Currently contains 8,862 rows and 3 columns, with information on:

**Target** Target identification; Galaxy-ID number

**HV** Weighted mean Heliocentric rest frame velocity

**Name** The name of the galaxy

## Source

https://vizier.u-strasbg.fr/viz-bin/VizieR-3?-source=J/AJ/137/3100/stars&-out.max=50&-out.form=HTML%20Table&-out.add=_r&-out.add=_RAJ,_DEJ&-out.add=_RA%2a-c.eq,_DE%2a-c.eq&-sort=_r&-oc.form=sexa

---

`BVdk_contrast` *Contrast as defined in Bordes & Vandekerkhove (2010)*

---

## Description

Compute the contrast as defined in Bordes & Vandekerkhove (2010) (see below in section 'Details'), needed for optimization purpose. Remind that one considers an admixture model with symmetric unknown density, i.e. l(x) = p*f(x-mu) + (1-p)*g(x), where l denotes the probability density function (pdf) of the mixture with known component pdf g, p is the unknown mixture weight, f relates to the unknown symmetric component pdf f, and mu is the location shift parameter.

## Usage

```
BVdk_contrast(param, data, h, comp.dist, comp.param)
```

## Arguments

| | |
|---|---|
| `param` | Numeric vector of two elements, corresponding to the two parameters (first the unknown component weight, and then the location shift parameter of the symmetric unknown component distribution). |
| `data` | Numeric vector of observations following the admixture model given by the pdf l. |
| `h` | Width of the window used in the kernel estimations. |
| `comp.dist` | A list with two elements corresponding to component distributions (specified with R native names for these distributions) involved in the admixture model. Unknown elements must be specified as 'NULL' objects, e.g. when 'f' is unknown: list(f=NULL, g='norm'). |
| `comp.param` | A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Unknown elements must be specified as 'NULL' objects, e.g. if 'f' is unknown: list(f=NULL, g=list(mean=0,sd=1)). |

## Details

The contrast is defined in Bordes, L. and Vandekerkhove, P. (2010); Semiparametric two-component mixture model when a component is known: an asymptotically normal estimator; Math. Meth. Stat.; 19, pp. 22–41.

## Value

The value of the contrast.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
comp.dist <- list(f = 'norm', g = 'norm')
comp.param <- list(f = list(mean = 3, sd = 0.5), g = list(mean = 0, sd = 1))
data1 <- rsimmix(n = 1000, unknownComp_weight = 0.6, comp.dist, comp.param)[['mixt.data']]
## Compute the contrast value for some given parameter vector in real-life framework:
comp.dist <- list(f = NULL, g = 'norm')
comp.param <- list(f = NULL, g = list(mean = 0, sd = 1))
BVdk_contrast(c(0.3,2), data1, density(data1)$bw, comp.dist, comp.param)
```

BVdk_contrast_gradient *Gradient of the contrast as defined in Bordes & Vandekerkhove (2010)*

## Description

Compute the gradient of the contrast as defined in Bordes & Vandekerkhove (2010) (see below in section 'Details'), needed for optimization purpose. Remind that one considers an admixture model, i.e. l = p*f + (1-p)*g ; where l denotes the probability density function (pdf) of the mixture with known component pdf g, p is the unknown mixture weight, and f relates to the unknown symmetric component pdf f.

## Usage

```
BVdk_contrast_gradient(param, data, h, comp.dist, comp.param)
```

## Arguments

| | |
|---|---|
| `param` | A numeric vector with two elements corresponding to the parameters to be estimated. First the unknown component weight, and second the location shift parameter of the symmetric unknown component distribution. |
| `data` | A vector of observations following the admixture model given by the pdf l. |
| `h` | The window width used in the kernel estimations. |
| `comp.dist` | A list with two elements corresponding to component distributions (specified with R native names for these distributions) involved in the admixture model. Unknown elements must be specified as 'NULL' objects, e.g. when 'f' is unknown: list(f=NULL, g='norm'). |
| `comp.param` | A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Unknown elements must be specified as 'NULL' objects, e.g. if 'f' is unknown: list(f=NULL, g=list(mean=0,sd=1)). |

## Details

The contrast is defined in Bordes, L. and Vandekerkhove, P. (2010); Semiparametric two-component mixture model when a component is known: an asymptotically normal estimator; Math. Meth. Stat.; 19, pp. 22–41.

## Value

A numeric vector composed of the two partial derivatives w.r.t. the two parameters on which to optimize the contrast.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
comp.dist <- list(f = 'norm', g = 'norm')
comp.param <- list(f = list(mean = 3, sd = 0.5), g = list(mean = 0, sd = 1))
data1 <- rsimmix(n = 1000, unknownComp_weight = 0.6, comp.dist, comp.param)[['mixt.data']]
## Compute the contrast gradient for some given parameter vector in real-life framework:
comp.dist <- list(f = NULL, g = 'norm')
comp.param <- list(f = NULL, g = list(mean = 0, sd = 1))
BVdk_contrast_gradient(c(0.3,2), data1, density(data1)$bw, comp.dist, comp.param)
```

---

# BVdk_estimParam — *Estimation of the parameters in a two-component admixture model with symmetric unknown density*

---

## Description

Estimation of the two parameters (mixture weight as well as location shift) in the admixture model with pdf: l(x) = p*f(x-mu) + (1-p)*g(x), x in R, where g is the known component, p is the proportion and f is the unknown component with symmetric density. The localization shift parameter is thus denoted mu, and the component weight p. See 'Details' below for further information.

## Usage

```
BVdk_estimParam(
  data,
  method = c("L-BFGS-B", "Nelder-Mead"),
  comp.dist,
  comp.param
)
```

## Arguments

| | |
|---|---|
| `data` | The observed sample under study. |
| `method` | The method used throughout the optimization process, either 'L-BFGS-B' or 'Nelder-Mead' (see ?optim). |
| `comp.dist` | A list with two elements corresponding to component distributions (specified with R native names for these distributions) involved in the admixture model. Unknown elements must be specified as 'NULL' objects, e.g. when 'f' is unknown: list(f=NULL, g='norm'). |
| `comp.param` | A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Unknown elements must be specified as 'NULL' objects, e.g. if 'f' is unknown: list(f=NULL, g=list(mean=0,sd=1)). |

### Details

Parameters are estimated by minimization of the contrast function, where the contrast is defined in Bordes, L. and Vandekerkhove, P. (2010); Semiparametric two-component mixture model when a component is known: an asymptotically normal estimator; Math. Meth. Stat.; 19, pp. 22–41.

### Value

A numeric vector with the two estimated parameters (proportion first, and then location shift).

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

### Examples

```
## Simulate data:
list.comp <- list(f = 'norm', g = 'norm')
list.param <- list(f = list(mean = 3, sd = 0.5),
                   g = list(mean = 0, sd = 1))
data1 <- rsimmix(n = 150, unknownComp_weight = 0.9, list.comp, list.param)[['mixt.data']]
## Perform the estimation of parameters in real-life:
list.comp <- list(f = NULL, g = 'norm')
list.param <- list(f = NULL, g = list(mean = 0, sd = 1))
BVdk_estimParam(data1, method = 'L-BFGS-B', list.comp, list.param)
```

---

## BVdk_ML_varCov_estimators

*Maximum Likelihood estimation of the variance of the unknown density variance estimator in an admixture model*

---

### Description

Parametric estimation of the variance of the variance parameter in Bordes & Vandekerkhove (2010) setting, i.e. considering the admixture model with probability density function (pdf) l: l(x) = p*f(x-mu) + (1-p)*g, where g is the known component of the two-component mixture, p is the mixture proportion, f is the unknown component with symmetric density, and mu is the location shift parameter. The estimation of the variance of the variance related to the density f is made by maximum likelihood optimization through the information matrix, with the assumption that the unknown f is gaussian.

### Usage

```
BVdk_ML_varCov_estimators(data, hat_w, hat_loc, hat_var, comp.dist, comp.param)
```

## Arguments

| | |
|---|---|
| data | The observed sample under study. |
| hat_w | Estimate of the unknown component weight. |
| hat_loc | Estimate of the location shift parameter. |
| hat_var | Estimate of the variance of the symmetric density f, obtained by plugging-in the previous estimates. See 'Details' below for further information. |
| comp.dist | A list with two elements corresponding to component distributions (specified with R native names for these distributions) involved in the admixture model. Unknown elements must be specified as 'NULL' objects, e.g. when 'f' is unknown: list(f=NULL, g='norm'). |
| comp.param | A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Unknown elements must be specified as 'NULL' objects, e.g. if 'f' is unknown: list(f=NULL, g=list(mean=0,sd=1)). |

## Details

Plug-in strategy is defined in Pommeret, D. and Vandekerkhove, P. (2019); Semiparametric density testing in the contamination model; Electronic Journal of Statistics, 13, pp. 4743–4793. The variance of the estimator variance of the unknown density f is needed in a testing perspective, since included in the variance of the test statistic. Other details about the information matrix can be found in Bordes, L. and Vandekerkhove, P. (2010); Semiparametric two-component mixture model when a component is known: an asymptotically normal estimator; Math. Meth. Stat.; 19, pp. 22–41.

## Value

The variance of the estimator of the variance of the unknown component density f.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f = "norm", g = "norm")
list.param <- list(f = c(mean = 4, sd = 1), g = c(mean = 7, sd = 0.5))
sim.data <- rsimmix(n = 400, unknownComp_weight = 0.9, list.comp, list.param)$mixt.data
## Estimate mixture weight and location shift parameters in real-life:
list.comp <- list(f = NULL, g = "norm")
list.param <- list(f = NULL, g = c(mean = 7, sd = 0.5))
estim <- BVdk_estimParam(data = sim.data, method = "L-BFGS-B",
                         comp.dist = list.comp, comp.param = list.param)
## Estimation of the second-order moment of the known component distribution:
m2_knownComp <- mean(rnorm(n = 1000000, mean = 7, sd = 0.5)^2)
hat_s2 <- (1/estim[1]) * (mean(sim.data^2) - ((1-estim[1])*m2_knownComp)) - estim[2]^2
```

```
## Estimated variance of variance estimator related to the unknown symmetric component density:
BVdk_ML_varCov_estimators(data = sim.data, hat_w = estim[1], hat_loc = estim[2],
                        hat_var = hat_s2, comp.dist = list.comp, comp.param = list.param)
```

---

`BVdk_varCov_estimators`

*Estimation of the variance of the estimators in admixture models with symmetric unknown density*

---

## Description

Semiparametric estimation of the variance of the estimators, i.e. the mixture weight p and the location shift parameter mu considering the admixture model with probability density function l: l(x) = p*f(x-mu) + (1-p)*g(x), x in R, where g is the known component of the two-component mixture, p is the unknown proportion, f is the unknown component density and mu is the location shift. See 'Details' below for more information.

## Usage

```
BVdk_varCov_estimators(data, loc, p, comp.dist, comp.param)
```

## Arguments

| | |
|---|---|
| `data` | The observed sample under study. |
| `loc` | The estimated location shift parameter, related to the unknown symmetric density. |
| `p` | The estimated unknown component weight. |
| `comp.dist` | A list with two elements corresponding to component distributions (specified with R native names for these distributions) involved in the admixture model. Unknown elements must be specified as 'NULL' objects, e.g. when 'f' is unknown: list(f=NULL, g='norm'). |
| `comp.param` | A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Unknown elements must be specified as 'NULL' objects, e.g. if 'f' is unknown: list(f=NULL, g=list(mean=0,sd=1)). |

## Details

See formulas pp.28–30 in Appendix of Bordes, L. and Vandekerkhove, P. (2010); Semiparametric two-component mixture model when a component is known: an asymptotically normal estimator; Math. Meth. Stat.; 19, pp. 22–41.

## Value

A list containing 1) the variance-covariance matrix of the estimators (assessed at the specific time points 'u' and 'v' such that u=v=mean(data)); 2) the variance of the mixture weight estimator; 3) the variance of the location shift estimator; 4) the variance of the unknown component cumulative distribution function at points 'u' and 'v' (useless for most of applications, explaining why 'u' and 'v' are set equal to mean(data) by default, with no corresponding arguments here).

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f = 'norm', g = 'norm')
list.param <- list(f = c(mean = 4, sd = 1), g = c(mean = 7, sd = 0.5))
sim.data <- rsimmix(n=140, unknownComp_weight=0.9, comp.dist=list.comp, comp.param=list.param)
## Estimate the location shift and mixture weight parameters in real-life setting:
list.comp <- list(f = NULL, g = 'norm')
list.param <- list(f = NULL, g = c(mean = 7, sd = 0.5))
estimators <- BVdk_estimParam(data = sim.data[['mixt.data']], method = "L-BFGS-B",
                              comp.dist = list.comp, comp.param = list.param)
## Estimate the variance of the two estimators (first mixture weight, then location shift):
BVdk_varCov_estimators(data = sim.data[['mixt.data']], loc = estimators[2], p = estimators[1],
                       comp.dist = list.comp, comp.param = list.param)
```

---

# decontamin_cdf_unknownComp

*Provide the decontaminated cumulative distribution function (CDF) of the unknown component in an admixture model*

---

## Description

Estimate the decontaminated CDF of the unknown component in the admixture model under study, after inversion of the admixture cumulative distribution function. Recall that an admixture model follows the cumulative distribution function (CDF) L, where L = p*F + (1-p)*G, with g a known CDF and p and f unknown quantities.

## Usage

```
decontamin_cdf_unknownComp(sample1, comp.dist, comp.param, estim.p)
```

## Arguments

| | |
|---|---|
| sample1 | Observations of the sample under study. |

`comp.dist` A list with two elements corresponding to the component distributions (specified with R native names for these distributions) involved in the admixture model. The two elements refer to the unknown and known components of the admixture model, If there are unknown elements, they must be specified as 'NULL' objects (e.g. 'comp.dist' could be set to list(f1=NULL, g1='norm')).

`comp.param` A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two elements refer to the parameters of unknown and known components of the admixture model. If there are unknown elements, they must be specified as 'NULL' objects (e.g. 'comp.param' could be set to list(f1=NULL, g1=list(mean=0,sd=1))).

`estim.p` The estimated weight of the unknown component distribution, related to the proportion of the unknown component in the admixture model studied.

## Details

The decontaminated CDF is obtained by inverting the admixture CDF, given by L = p*F + (1-p)*G, to isolate the unknown component F after having estimated p. This means that F = (1/hat(p)) * (hat(L)-(1-p)*G).

## Value

The decontaminated CDF F of the admixture model, as an of class 'stepfun' (step function).

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
####### Continuous support:
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 3, sd = 0.5), g2 = list(mean = 5, sd = 2))
sample1 <- rsimmix(n=3500, unknownComp_weight=0.5, comp.dist = list(list.comp$f1,list.comp$g1),
                                        comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=3000, unknownComp_weight=0.7, comp.dist = list(list.comp$f2,list.comp$g2),
                                        comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'norm',
                  f2 = NULL, g2 = 'norm')
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1),
                   f2 = NULL, g2 = list(mean = 5, sd = 2))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
## Determine the decontaminated version of the unknown CDF by inversion:
decontamin_cdf_unknownComp(sample1 = sample1[['mixt.data']],
                           comp.dist = list.comp[1:2], comp.param = list.param[1:2],
```

```
                          estim.p = estimate$prop.estim[1])
####### Countable discrete support:
list.comp <- list(f1 = 'pois', g1 = 'pois',
                  f2 = 'pois', g2 = 'pois')
list.param <- list(f1 = list(lambda = 3), g1 = list(lambda = 2),
                   f2 = list(lambda = 3), g2 = list(lambda = 4))
sample1 <- rsimmix(n=6000, unknownComp_weight=0.6, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=4500, unknownComp_weight=0.8, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'pois',
                  f2 = NULL, g2 = 'pois')
list.param <- list(f1 = NULL, g1 = list(lambda = 2),
                   f2 = NULL, g2 = list(lambda = 4))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
decontamin_cdf_unknownComp(sample1 = sample1[['mixt.data']],
                          comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                          estim.p = estimate$prop.estim[1])
####### Finite discrete support:
list.comp <- list(f1 = 'multinom', g1 = 'multinom',
                  f2 = 'multinom', g2 = 'multinom')
list.param <- list(f1 = list(size=1, prob=c(0.3,0.4,0.3)), g1 = list(size=1, prob=c(0.6,0.3,0.1)),
             f2 = list(size=1, prob=c(0.3,0.4,0.3)), g2 = list(size=1, prob=c(0.2,0.6,0.2)))
sample1 <- rsimmix(n=8000, unknownComp_weight=0.6, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=6000, unknownComp_weight=0.8, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
list.comp <- list(f1 = NULL, g1 = 'multinom',
                  f2 = NULL, g2 = 'multinom')
list.param <- list(f1 = NULL, g1 = list(size=1, prob=c(0.6,0.3,0.1)),
                   f2 = NULL, g2 = list(size=1, prob=c(0.2,0.6,0.2)))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
decontamin_cdf_unknownComp(sample1 = sample1[['mixt.data']],
                          comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                          estim.p = estimate$prop.estim[1])
```

---

`decontamin_density_unknownComp`

*Provide the decontaminated density of the unknown component in an admixture model*

---

## Description

Estimate the decontaminated density of the unknown component in the admixture model under study, after inversion of the admixture cumulative distribution function. Recall that an admixture model follows the cumulative distribution function (CDF) L, where L = p*F + (1-p)*G, with g a known CDF and p and f unknown quantities.

## Usage

```
decontamin_density_unknownComp(sample1, comp.dist, comp.param, estim.p)
```

## Arguments

| | |
|---|---|
| sample1 | Observations of the first sample under study. |
| comp.dist | A list with two elements corresponding to the component distributions (specified with R native names for these distributions) involved in the admixture model. The two elements refer to the unknown and known components of the admixture model, If there are unknown elements, they must be specified as 'NULL' objects (e.g. 'comp.dist' could be set to list(f1=NULL, g1='norm')). |
| comp.param | A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two elements refer to the parameters of unknown and known components of the admixture model. If there are unknown elements, they must be specified as 'NULL' objects (e.g. 'comp.param' could be set to list(f1=NULL, g1=list(mean=0,sd=1))). |
| estim.p | The estimated weight of the unknown component distribution, related to the proportion of the unknown component in the admixture model studied. |

## Details

The decontaminated density is obtained by inverting the admixture density, given by l = p*f + (1-p)*g, to isolate the unknown component f after having estimated p.

## Value

A list containing the decontaminated density of the admixture model (of class 'function'), and the support of the observations (either discrete or continuous).

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 3, sd = 0.5), g2 = list(mean = 5, sd = 2))
sample1 <- rsimmix(n=8000, unknownComp_weight=0.5, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=7000, unknownComp_weight=0.7, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'norm',
                  f2 = NULL, g2 = 'norm')
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1),
```

```
                  f2 = NULL, g2 = list(mean = 5, sd = 2))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
## Determine the decontaminated version of the unknown density by inversion:
decontamin_density_unknownComp(sample1 = sample1[['mixt.data']],
                               comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                               estim.p = estimate$prop.estim[1])
####### Discrete support:
list.comp <- list(f1 = 'pois', g1 = 'pois',
                  f2 = 'pois', g2 = 'pois')
list.param <- list(f1 = list(lambda = 3), g1 = list(lambda = 2),
                   f2 = list(lambda = 3), g2 = list(lambda = 4))
sample1 <- rsimmix(n=7000, unknownComp_weight=0.6, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=6000, unknownComp_weight=0.8, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'pois',
                  f2 = NULL, g2 = 'pois')
list.param <- list(f1 = NULL, g1 = list(lambda = 2),
                   f2 = NULL, g2 = list(lambda = 4))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
## Determine the decontaminated version of the unknown density by inversion:
decontamin_density_unknownComp(sample1 = sample1[['mixt.data']],
                               comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                               estim.p = estimate$prop.estim[1])
####### Finite discrete support:
list.comp <- list(f1 = 'multinom', g1 = 'multinom',
                  f2 = 'multinom', g2 = 'multinom')
list.param <- list(f1 = list(size=1, prob=c(0.3,0.4,0.3)), g1 = list(size=1, prob=c(0.6,0.3,0.1)),
             f2 = list(size=1, prob=c(0.3,0.4,0.3)), g2 = list(size=1, prob=c(0.2,0.6,0.2)))
sample1 <- rsimmix(n=12000, unknownComp_weight=0.6, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=10000, unknownComp_weight=0.8, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'multinom',
                  f2 = NULL, g2 = 'multinom')
list.param <- list(f1 = NULL, g1 = list(size=1, prob=c(0.6,0.3,0.1)),
                   f2 = NULL, g2 = list(size=1, prob=c(0.2,0.6,0.2)))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
## Determine the decontaminated version of the unknown density by inversion:
decontamin_density_unknownComp(sample1 = sample1[['mixt.data']],
                               comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                               estim.p = estimate$prop.estim[1])
```

---

`detect_support_type` *Detect the support of the random variables under study*

---

### Description

Given one or two sets of observations (two samples), the function provides with the most plausible type of support for the underlying random variables to be studied. Basically, if less than 3 percent of the observations have different values, we consider that the support is discrete. Otherwise, we consider it as a continuous support.

### Usage

```
detect_support_type(sample1, sample2 = NULL)
```

### Arguments

| | |
|---|---|
| sample1 | The first sample of observations under study. |
| sample2 | The second sample of observations under study. |

### Value

The type of support, either discrete or continuous.

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

### Examples

```
## Simulate the two mixture samples:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 1, sd = 0.1), g2 = list(mean = 5, sd = 2))
sample1 <- rsimmix(n=1500, unknownComp_weight=0.5, comp.dist = list(list.comp$f1,list.comp$g1),
                   comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=2000, unknownComp_weight=0.7, comp.dist = list(list.comp$f2,list.comp$g2),
                   comp.param=list(list.param$f2,list.param$g2))
## Test the type of support:
detect_support_type(sample1[['mixt.data']], sample2[['mixt.data']])
```

---

## estimVarCov_empProcess

*Variance-covariance matrix of the empirical process in an admixture model*

---

### Description

Estimate the variance-covariance matrix of some given empirical process, based on the Donsker correlation. Compute Donsker correlation between two time points (x,y) for some given empirical process with R code (another implementation in C++ is also available to speed up this computation).

## Usage

```
estimVarCov_empProcess(
  x,
  y,
  obs.data,
  known.p = NULL,
  comp.dist = NULL,
  comp.param = NULL
)
```

## Arguments

| | |
|---|---|
| x | First time point considered for the computation of the correlation given the empirical process. |
| y | Second time point considered for the computation of the correlation given the same empirical process. |
| obs.data | Sample that permits to estimate the cumulative distribution function (cdf). |
| known.p | NULL by default (only useful to compute the exact Donsker correlation). The component weight dedicated to the unknown mixture component if available (in case of simulation studies) |
| comp.dist | NULL by default (only useful to compute the exact Donsker correlation). Otherwise, a list with two elements corresponding to component distributions (specified with R native names for these distributions) involved in the admixture model. All elements must be specified, for instance list(f='norm', g='norm'). |
| comp.param | NULL by default (only useful to compute the exact Donsker correlation). Otherwise, a list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. All elements must be specified, for instance list(f=NULL, g=list(mean=0,sd=1)). |

## Value

The estimated variance-covariance matrix.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm')
list.param <- list(f1 = list(mean = 12, sd = 0.4),
                   g1 = list(mean = 16, sd = 0.7))
obs.data <- rsimmix(n=2500, unknownComp_weight=0.5, comp.dist=list.comp, comp.param= list.param)
## Compute the variance-covariance matrix of the corresponding empirical process:
t <- seq(from = min(obs.data$mixt.data), to = max(obs.data$mixt.data), length = 50)
S2 <- sapply(t, function(s1) {
```

```
                sapply(t, function(s2) {
                 estimVarCov_empProcess(x = s1, y = s2, obs.data = obs.data$mixt.data) })
                })
lattice::wireframe(S2)
```

---

gaussianity_test *One-sample gaussianity test in admixture models using Bordes and Vandekerkhove estimation method*

---

## Description

Perform the hypothesis test to know whether the unknown mixture component is gaussian or not, knowing that the known one has support on the real line (R). However, the case of non-gaussian known component can be overcome thanks to the basic transformation by cdf. Recall that an admixture model has probability density function (pdf) l = p*f + (1-p)*g, where g is the known pdf and l is observed (others are unknown). Requires optimization (to estimate the unknown parameters) as defined by Bordes & Vandekerkhove (2010), which means that the unknown mixture component must have a symmetric density.

## Usage

```
gaussianity_test(
  sample1,
  comp.dist,
  comp.param,
  K = 3,
  lambda = 0.2,
  support = c("Real", "Integer", "Positive", "Bounded.continuous")
)
```

## Arguments

| | |
|---|---|
| sample1 | Observed sample with mixture distribution given by l = p*f + (1-p)*g, where f and p are unknown and g is known. |
| comp.dist | List with two elements corresponding to the component distributions involved in the admixture model. Unknown elements must be specified as 'NULL' objects. For instance if 'f' is unknown: list(f = NULL, g = 'norm'). |
| comp.param | List with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R names for distributions. Unknown elements must be specified as 'NULL' objects (e.g. if 'f' is unknown: list(f=NULL, g=list(mean=0,sd=1)). |
| K | Number of coefficients considered for the polynomial basis expansion. |
| lambda | Rate at which the normalization factor is set in the penalization rule for model selection (in ]0,1/2[). See 'Details' below. |
| support | Support of the densities under consideration, useful to choose the polynomial orthonormal basis. One of 'Real', 'Integer', 'Positive', or 'Bounded.continuous'. |

## Details

See the paper 'False Discovery Rate model Gaussianity test' (Pommeret & Vanderkerkhove, 2017).

## Value

A list of 6 elements, containing: 1) the rejection decision; 2) the p-value of the test; 3) the test statistic; 4) the variance-covariance matrix of the test statistic; 5) the selected rank for testing; and 6) a list of the estimates (unknown component weight 'p', shift location parameter 'mu' and standard deviation 's' of the symmetric unknown distribution).

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
####### Under the null hypothesis H0.
## Parameters of the gaussian distribution to be tested:
list.comp <- list(f = "norm", g = "norm")
list.param <- list(f = c(mean = 2, sd = 0.5),
                   g = c(mean = 0, sd = 1))
## Simulate and plot the data at hand:
obs.data <- rsimmix(n = 150, unknownComp_weight = 0.9, comp.dist = list.comp,
                    comp.param = list.param)[['mixt.data']]
plot(density(obs.data))
## Performs the test:
list.comp <- list(f = NULL, g = "norm")
list.param <- list(f = NULL, g = c(mean = 0, sd = 1))
gaussianity_test(sample1 = obs.data, comp.dist = list.comp, comp.param = list.param,
                      K = 3, lambda = 0.1, support = 'Real')
```

---

# IBM_empirical_contrast

*Empirical computation of the contrast in the Inversion - Best Matching (IBM) method*

---

## Description

Defines the empirical version of the contrast in the IBM method, to be minimized in the optimization process. For further details about the contrast definition, see 'Details' below.

## Usage

```
IBM_empirical_contrast(
  par,
  fixed.p.X = NULL,
  sample1,
```

```
  sample2,
  G,
  comp.dist,
  comp.param
)
```

## Arguments

| | |
|---|---|
| par | Numeric vector with two elements, corresponding to the two parameter values at which to compute the contrast. In practice the component weights for the two admixture models. |
| fixed.p.X | Arbitrary value chosen by the user for the component weight related to the unknown component of the first admixture model. Only useful for optimization when the known components of the two models are identical (G1=G2, leading to unidimensional optimization). |
| sample1 | Observations of the first sample under study. |
| sample2 | Observations of the second sample under study. |
| G | Distribution on which to integrate when calculating the contrast. |
| comp.dist | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |
| comp.param | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)). |

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

The empirical contrast value evaluated at parameter values.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 1, sd = 0.1), g2 = list(mean = 5, sd = 2))
sample1 <- rsimmix(n=1500, unknownComp_weight=0.5, comp.dist = list(list.comp$f1,list.comp$g1),
                                        comp.param = list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=2000, unknownComp_weight=0.7, comp.dist = list(list.comp$f2,list.comp$g2),
                                        comp.param = list(list.param$f2,list.param$g2))
## Create the distribution on which the contrast will be integrated:
G <- stats::rnorm(n = 1000, mean = sample(c(sample1[['mixt.data']], sample2[['mixt.data']]),
                                          size = 1000, replace = TRUE),
                  sd = density(c(sample1[['mixt.data']], sample2[['mixt.data']]))$bw)
## Compute the empirical contrast at parameters (p1,p2) = (0.2,0.7) in a real-life setting:
list.comp <- list(f1 = NULL, g1 = 'norm',
                  f2 = NULL, g2 = 'norm')
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1),
                   f2 = NULL, g2 = list(mean = 5, sd = 2))
IBM_empirical_contrast(par = c(0.2,0.7), fixed.p.X = NULL, sample1 = sample1[['mixt.data']],
       sample2= sample2[['mixt.data']], G=G, comp.dist = list.comp, comp.param = list.param)
```

---

IBM_estimProp *Estimate the weights related to the proportions of the unknown components of the two admixture models*

---

## Description

Estimate the component weights from the Inversion - Best Matching (IBM) method, related to the two admixture models with respective probability density function (pdf) l1 and l2, such that: l1 = p1*f1 + (1-p1)*g1 and l2* = *p2*f2 + (1-p2)*g2, where g1 and g2 are the known component densities. For further details about IBM approach, see 'Details' below.

## Usage

```
IBM_estimProp(
  sample1,
  sample2,
  known.prop = NULL,
  comp.dist = NULL,
  comp.param = NULL,
  with.correction = TRUE,
  n.integ = 1000
)
```

## Arguments

| | |
|---|---|
| sample1 | Observations of the first sample under study. |
| sample2 | Observations of the second sample under study. |
| known.prop | (optional) Numeric vector with two elements, respectively the component weight for the unknown component in the first and in the second samples. |
| comp.dist | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |
| comp.param | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)). |
| with.correction | Boolean indicating whether the solution (estimated proportions) should be adjusted or not (with the constant determined thanks to the exact proportion, usually unknown except in case of simulations). |
| n.integ | Number of data points generated for the distribution on which to integrate. |

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

A list with the two estimates of the component weights for each of the admixture model, plus that of the theoretical model if specified.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
##### On a simulated example to see whether the true parameters are well estimated.
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 3, sd = 0.5), g2 = list(mean = 5, sd = 2))
```

```
sample1 <- rsimmix(n=1500, unknownComp_weight=0.5, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=2000, unknownComp_weight=0.7, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weights of the two admixture models (provide hat(theta)_n and theta^c):
estim <- IBM_estimProp(sample1 = sample1[['mixt.data']], sample2 = sample2[['mixt.data']],
                  known.prop = c(0.5,0.7), comp.dist = list.comp, comp.param = list.param,
                       with.correction = FALSE, n.integ = 1000)
estim[['prop.estim']]
estim[['theo.prop.estim']]
##### On a real-life example (unknown component densities, unknown mixture weights).
list.comp <- list(f1 = NULL, g1 = 'norm',
                  f2 = NULL, g2 = 'norm')
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1),
                   f2 = NULL, g2 = list(mean = 5, sd = 2))
## Estimate the mixture weights of the two admixture models (provide only hat(theta)_n):
estim <- IBM_estimProp(sample1 = sample1[['mixt.data']], sample2 = sample2[['mixt.data']],
                      known.prop = NULL, comp.dist = list.comp, comp.param = list.param,
                       with.correction = FALSE, n.integ = 1000)
estim[['prop.estim']]
estim[['theo.prop.estim']]
```

---

`IBM_estimVarCov_gaussVect` *Nonparametric estimation of the variance-covariance matrix of the gaussian vector in IBM approach*

---

## Description

Estimate the variance-covariance matrix of the gaussian vector at point 'z', considering the use of Inversion - Best Matching (IBM) method to estimate the model parameters in two-sample admixture models. Recall that the two admixture models have respective probability density functions (pdf) l1 and l2, such that: l1 = p1*f1 + (1-p1)*g1 and l2 = p2*f2 + (1-p2)*g2, where g1 and g2 are the known component densities. Further information for the IBM approach are given in 'Details' below.

## Usage

```
IBM_estimVarCov_gaussVect(
  x,
  y,
  estim.obj,
  fixed.p1 = NULL,
  known.p = NULL,
  sample1,
  sample2,
  min_size = NULL,
  comp.dist = NULL,
  comp.param = NULL
)
```

## Arguments

| | |
|---|---|
| x | Time point at which the first (related to the first parameter) underlying empirical process is looked through. |
| y | Time point at which the second (related to the second parameter) underlying empirical process is looked through. |
| estim.obj | Object obtained from the estimation of the component weights related to the proportions of the unknown component in each of the two admixture models. |
| fixed.p1 | Arbitrary value chosen by the user for the component weight related to the unknown component of the first admixture model. Only useful for optimization when the known components of the two models are identical (G1=G2, leading to unidimensional optimization). |
| known.p | (optional, NULL by default) Numeric vector with two elements, the known (true) mixture weights. |
| sample1 | Observations of the first sample under study. |
| sample2 | Observations of the second sample under study. |
| min_size | (optional, NULL by default) in the k-sample case, useful to provide the minimal size among all samples (needed to take into account the correction factor in variance-covariance assessment). Otherwise, useless. |
| comp.dist | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |
| comp.param | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)). |

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

The estimated variance-covariance matrix of the gaussian vector Z = (hat(p1),(hat(p2),Dn(z)), at location '(x,y)'.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
######## Analysis by simulated data:
## Simulate Gamma - Exponential admixtures :
list.comp <- list(f1 = "gamma", g1 = "exp",
                  f2 = "gamma", g2 = "exp")
list.param <- list(f1 = list(shape = 2, scale = 3), g1 = list(rate = 1/3),
                   f2 = list(shape = 2, scale = 3), g2 = list(rate = 1/5))
X.sim <- rsimmix(n=400, unknownComp_weight=0.8, comp.dist = list(list.comp$f1,list.comp$g1),
                 comp.param = list(list.param$f1, list.param$g1))$mixt.data
Y.sim <- rsimmix(n=350, unknownComp_weight=0.9, comp.dist = list(list.comp$f2,list.comp$g2),
                 comp.param = list(list.param$f2, list.param$g2))$mixt.data
## Real-life setting:
list.comp <- list(f1 = NULL, g1 = "exp",
                  f2 = NULL, g2 = "exp")
list.param <- list(f1 = NULL, g1 = list(rate = 1/3),
                   f2 = NULL, g2 = list(rate = 1/5))
## Estimate the unknown component weights in the two admixture models:
estim <- IBM_estimProp(sample1 =X.sim, sample2 =Y.sim, known.prop = NULL, comp.dist = list.comp,
                       comp.param = list.param, with.correction = FALSE, n.integ = 1000)
IBM_estimVarCov_gaussVect(x = mean(X.sim), y = mean(Y.sim), estim.obj = estim,
                          fixed.p1 = estim[["p.X.fixed"]], known.p = NULL, sample1=X.sim,
                          sample2 = Y.sim, min_size = NULL,
                          comp.dist = list.comp, comp.param = list.param)
```

---

`IBM_gap` *Difference between the unknown empirical cumulative distribution functions in two admixture models*

---

## Description

Compute the 'gap' between two unknown cumulative distribution functions (ecdf) at some given point, in admixture models with probability distribution function (pdf) given by l where l = p*f + (1-p)*g. Uses the inversion method to do so, i.e. f = (1/p) (l - (1-p)*g), where g represents the known component of the admixture model and p is the unknown proportion of the unknown component. Therefore, compute: D(z,L1,L2,p1,p2) = F1(z,L1,p1) - F2(z,L2,p2) This measure should be integrated over some domain to compute the global discrepancy, see further information in 'Details' below.

## Usage

```
IBM_gap(z, par, fixed.p1 = NULL, sample1, sample2, comp.dist, comp.param)
```

## Arguments

| | |
|---|---|
| `z` | the point at which the difference between both unknown (estimated) component distributions is computed. |

| | |
|---|---|
| par | Numeric vector with two elements, corresponding to the weights of the unknown component for the two admixture models. |
| fixed.p1 | (optional, NULL by default) Arbitrary value chosen by the user for the component weight related to the unknown component of the first admixture model. Only useful for optimization when the known components of the two models are identical (G1=G2, leading to unidimensional optimization). |
| sample1 | Observations of the first sample under study. |
| sample2 | Observations of the second sample under study. |
| comp.dist | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |
| comp.param | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)). |

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

the gap evaluated at the specified point between the unknown components of the two observed samples.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 1, sd = 0.1), g2 = list(mean = 5, sd = 2))
sample1 <- rsimmix(n=1500, unknownComp_weight=0.5, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=2000, unknownComp_weight=0.7, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
IBM_gap(z = 2.8, par = c(0.3,0.6), fixed.p1 = NULL, sample1 = sample1[['mixt.data']],
```

```
sample2 = sample2[['mixt.data']], comp.dist = list.comp, comp.param = list.param)
```

---

`IBM_greenLight_criterion` *Green-light criterion to decide whether to perform full equality test between unknown components between two admixture models*

---

## Description

Indicate whether there is need to perform the statistical test of equality between unknown components when comparing the unknown components of two samples following admixture models. Based on the IBM approach, see more in 'Details' below.

## Usage

```
IBM_greenLight_criterion(
  estim.obj,
  sample1,
  sample2,
  comp.dist = NULL,
  comp.param = NULL,
  min_size = NULL,
  alpha = 0.05
)
```

## Arguments

| | |
|---|---|
| `estim.obj` | Object obtained from the estimation of the component weights related to the proportions of the unknown component in each of the two admixture models studied. |
| `sample1` | Observations of the first sample under study. |
| `sample2` | Observations of the second sample under study. |
| `comp.dist` | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |
| `comp.param` | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture |

model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)).

`min_size` (optional, NULL by default) In the k-sample case, useful to provide the minimal size among all samples (needed to take into account the correction factor for variance-covariance assessment). Otherwise, useless.

`alpha` Confidence level at which the criterion is assessed (used to compute the confidence bands of the estimators of the unknown component weights).

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

A boolean indicating whether it is useful or useless to tabulate the contrast distribution in order to answer the testing problem (f1 = f2).

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 3, sd = 0.5), g2 = list(mean = 5, sd = 2))
sample1 <- rsimmix(n=550, unknownComp_weight=0.7, comp.dist = list(list.comp$f1,list.comp$g1),
                                                  comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=450, unknownComp_weight=0.8, comp.dist = list(list.comp$f2,list.comp$g2),
                                                  comp.param=list(list.param$f2,list.param$g2))
## Estimate the unknown component weights in the two admixture models in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'norm',
                  f2 = NULL, g2 = 'norm')
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1),
                   f2 = NULL, g2 = list(mean = 5, sd = 2))
estim <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], known.prop = NULL,
                       comp.dist = list.comp, comp.param = list.param,
                       with.correction = FALSE, n.integ = 1000)
IBM_greenLight_criterion(estim.obj = estim, sample1 = sample1[['mixt.data']],
                         sample2 = sample2[['mixt.data']], comp.dist = list.comp,
                         comp.param = list.param, min_size = NULL, alpha = 0.05)
```

`IBM_hessian_contrast` *Hessian matrix of the contrast function in the Inversion - Best Matching (IBM) method*

## Description

Compute the hessian matrix of the contrast as defined in the IBM approach, at point (p1,p2). Here, based on two samples following admixture models, where we recall that admixture models have probability distribution function (pdf) given by l where l = p*f + (1-p)*g, where g represents the only known quantity and l is the pdf of the observed sample. See 'Details' below for further information about the definition of the contrast.

## Usage

```
IBM_hessian_contrast(
  par,
  fixed.p1 = NULL,
  known.p = NULL,
  sample1,
  sample2,
  G,
  comp.dist = NULL,
  comp.param = NULL
)
```

## Arguments

| | |
|---|---|
| `par` | Numeric vector with two elements (corresponding to the two unknown component weights) at which the hessian is computed. |
| `fixed.p1` | (optional, NULL by default) Arbitrary value chosen by the user for the component weight related to the unknown component of the first admixture model. Only useful for optimization when the known components of the two models are identical (G1=G2, leading to unidimensional optimization). |
| `known.p` | (optional, NULL by default) Numeric vector with two elements, the known (true) mixture weights. |
| `sample1` | Observations of the first sample under study. |
| `sample2` | Observations of the second sample under study. |
| `G` | Distribution on which to integrate when calculating the contrast. |
| `comp.dist` | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |

comp.param A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)).

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

the hessian matrix of the contrast.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 1, sd = 0.1), g2 = list(mean = 5, sd = 2))
sample1 <- rsimmix(n=1500, unknownComp_weight=0.5, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=2000, unknownComp_weight=0.7, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Define the distribution over which to integrate:
fit.all <- stats::density(x = c(sample1[['mixt.data']],sample2[['mixt.data']]))
G <- stats::rnorm(n = 1000, mean = sample(c(sample1[['mixt.data']], sample2[['mixt.data']]),
                                        size = 1000, replace = TRUE), sd = fit.all$bw)
## Evaluate the hessian matrix at point (p1,p2) = (0.3,0.6):
list.comp <- list(f1 = NULL, g1 = 'norm',
                  f2 = NULL, g2 = 'norm')
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1),
                   f2 = NULL, g2 = list(mean = 5, sd = 2))
IBM_hessian_contrast(par = c(0.3,0.6), fixed.p1 = NULL, known.p = NULL,
               sample1 = sample1[['mixt.data']],  sample2 = sample2[['mixt.data']], G = G,
                    comp.dist = list.comp, comp.param = list.param)
```

`IBM_k_samples_test` *Equality test of unknown component distributions in K admixture models, with IBM approach*

## Description

Test hypothesis on the unknown component of K (K > 1) admixture models using Inversion - Best Matching method. K-samples test of the unknown component distribution in admixture models using Inversion - Best Matching (IBM) method. Recall that we have K populations following admixture models, each one with probability density functions (pdf) l_k = p_k*f_k + (1-p_k)*g_k, where g_k is the known pdf and l_k corresponds to the observed sample. Perform the following hypothesis test: H0 : f_1 = ... = f_K against H1 : f_i differs from f_j (i different from j, and i,j in 1,...,K).

## Usage

```
IBM_k_samples_test(
  samples = NULL,
  sim_U = NULL,
  n_sim_tab = 100,
  min_size = NULL,
  comp.dist = NULL,
  comp.param = NULL,
  parallel = FALSE,
  n_cpu = 2
)
```

## Arguments

| | |
|---|---|
| `samples` | A list of the K samples to be studied, all following admixture distributions. |
| `sim_U` | (default to NULL) Random draws of the inner convergence part of the contrast as defined in the IBM approach (see 'Details' below). |
| `n_sim_tab` | Number of simulated gaussian processes when tabulating the inner convergence distribution in the IBM approach. |
| `min_size` | (default to NULL) Useful to provide the minimal size among all samples (needed to take into account the correction factor for the variance-covariance assessment). Automatically calculated if NULL. |
| `comp.dist` | A list with 2*K elements corresponding to the component distributions (specified with R native names for these distributions) involved in the K admixture models. Elements, grouped by 2, refer to the unknown and known components of each admixture model, If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows with K = 3: list(f1 = NULL, g1 = 'norm', f2 = NULL, g2 = 'norm', f3 = NULL, g3 = 'rnorm'). |

| | |
|---|---|
| comp.param | A list with 2*K elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Elements, grouped by 2, refer to the parameters of unknown and known components of each admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows (with K = 3): list(f1 = NULL, g1 = list(mean=0,sd=1), f2 = NULL, g2 = list(mean=3,sd=1.1), f3 = NULL, g3 = list(mean=-2,sd=0.6)). |
| parallel | (default to FALSE) Boolean indicating whether parallel computations are performed. |
| n_cpu | (default to 2) Number of cores used when parallelizing. |

## Details

See the paper at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

A list of ten elements, containing: 1) the rejection decision; 2) the test p-value; 3) the terms involved in the test statistic; 4) the test statistic value; 5) the selected rank (number of terms involved in the test statistic); 6) the value of the penalized test statistic; 7) a boolean indicating whether the applied penalty rule is that under the null H0; 8) the sorted contrast values; 9) the 95th-quantile of the contrast distribution; 10) the final terms of the statistic; and 11) the contrast matrix.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
####### Under the null hypothesis H0 (with K=3 populations):
## Specify the parameters of the mixture models for simulation:
list.comp <- list(f1 = "norm", g1 = "norm",
                  f2 = "norm", g2 = "norm",
                  f3 = "norm", g3 = "norm")
list.param <- list(f1 = list(mean = 0, sd = 1), g1 = list(mean = 2, sd = 0.7),
                   f2 = list(mean = 0, sd = 1), g2 = list(mean = 4, sd = 1.1),
                   f3 = list(mean = 0, sd = 1), g3 = list(mean = -3, sd = 0.8))
## Simulate the data:
sim1 <- rsimmix(n = 1000, unknownComp_weight = 0.8, comp.dist = list(list.comp$f1,list.comp$g1),
                comp.param = list(list.param$f1, list.param$g1))$mixt.data
sim2 <- rsimmix(n= 1300, unknownComp_weight = 0.6, comp.dist = list(list.comp$f2,list.comp$g2),
                comp.param = list(list.param$f2, list.param$g2))$mixt.data
sim3 <- rsimmix(n = 1100, unknownComp_weight = 0.7, comp.dist = list(list.comp$f3,list.comp$g3),
                comp.param = list(list.param$f3, list.param$g3))$mixt.data
## Back to the context of admixture models, where one mixture component is unknown:
list.comp <- list(f1 = NULL, g1 = "norm",
                  f2 = NULL, g2 = "norm",
                  f3 = NULL, g3 = "norm")
```

```
list.param <- list(f1 = NULL, g1 = list(mean = 2, sd = 0.7),
                   f2 = NULL, g2 = list(mean = 4, sd = 1.1),
                   f3 = NULL, g3 = list(mean = -3, sd = 0.8))
## Perform the 3-samples test:
IBM_k_samples_test(samples = list(sim1,sim2,sim3), sim_U= NULL, n_sim_tab = 20, min_size = NULL,
               comp.dist = list.comp, comp.param = list.param, parallel = TRUE, n_cpu = 2)

####### Now under the alternative H1:
list.comp <- list(f1 = "norm", g1 = "norm",
                  f2 = "norm", g2 = "norm",
                  f3 = "norm", g3 = "norm")
list.param <- list(f1 = list(mean = 0, sd = 1), g1 = list(mean = 2, sd = 0.7),
                   f2 = list(mean = 0, sd = 1), g2 = list(mean = 4, sd = 1.1),
                   f3 = list(mean = 2, sd = 0.7), g3 = list(mean = 3, sd = 0.8))
sim1 <- rsimmix(n = 3000, unknownComp_weight = 0.8, comp.dist = list(list.comp$f1,list.comp$g1),
                comp.param = list(list.param$f1, list.param$g1))$mixt.data
sim2 <- rsimmix(n= 3300, unknownComp_weight = 0.6, comp.dist = list(list.comp$f2,list.comp$g2),
                comp.param = list(list.param$f2, list.param$g2))$mixt.data
sim3 <- rsimmix(n = 3100, unknownComp_weight = 0.7, comp.dist = list(list.comp$f3,list.comp$g3),
                comp.param = list(list.param$f3, list.param$g3))$mixt.data
list.comp <- list(f1 = NULL, g1 = "norm",
                  f2 = NULL, g2 = "norm",
                  f3 = NULL, g3 = "norm")
list.param <- list(f1 = NULL, g1 = list(mean = 2, sd = 0.7),
                   f2 = NULL, g2 = list(mean = 4, sd = 1.1),
                   f3 = NULL, g3 = list(mean = 3, sd = 0.8))
IBM_k_samples_test(samples = list(sim1,sim2,sim3), sim_U= NULL, n_sim_tab = 20, min_size = NULL,
               comp.dist = list.comp, comp.param = list.param, parallel = TRUE, n_cpu = 2)
```

---

`IBM_tabul_stochasticInteg` *Distribution of the contrast in the Inversion - Best Matching (IBM) method*

---

## Description

Tabulate the distribution related to the inner convergence part of the contrast, by simulating trajectories of gaussian processes and applying some transformations. Useful to perform the test hypothesis then, by retrieving the (1-alpha)-quantile of interest. See 'Details' below and the cited paper therein for further information.

## Usage

```
IBM_tabul_stochasticInteg(
  n.sim = 200,
  n.varCovMat = 100,
  sample1 = NULL,
```

```
  sample2 = NULL,
  min_size = NULL,
  comp.dist = NULL,
  comp.param = NULL,
  parallel = FALSE,
  n_cpu = 2
)
```

## Arguments

| | |
|---|---|
| n.sim | Number of trajectories of simulated gaussian processes (number of random draws for tabulation). |
| n.varCovMat | Number of time points on which gaussian processes are simulated. |
| sample1 | Observations of the first sample under study. |
| sample2 | Observations of the second sample under study. |
| min_size | (default to NULL) In the k-sample case, useful to provide the minimal size among all samples. Otherwise, useless. |
| comp.dist | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |
| comp.param | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)). |
| parallel | (default to FALSE) Boolean to indicate whether parallel computations are performed (speed-up the tabulation). |
| n_cpu | (default to 2) Number of cores used for computations when parallelizing. |

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

A list with four elements, containing: 1) random draws of the quantity 'sample size times the empirical contrast', as defined in the IBM approach (see 'Details' above); 2) the estimated unknown component weights for the two admixture models; 3) the value of the quantity 'sample size times the empirical contrast'; 4) the sequence of points in the support that were used to evaluate the variance-covariance matrix of empirical processes.

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

### Examples

```
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 1, sd = 1), g1 = list(mean = 2, sd = 0.7),
                   f2 = list(mean = 1, sd = 1), g2 = list(mean = 3, sd = 1.2))
X.sim <- rsimmix(n=1000, unknownComp_weight=0.7, comp.dist = list(list.comp$f1,list.comp$g1),
                 comp.param = list(list.param$f1, list.param$g1))$mixt.data
Y.sim <- rsimmix(n=1200, unknownComp_weight=0.6, comp.dist = list(list.comp$f2,list.comp$g2),
                 comp.param = list(list.param$f2, list.param$g2))$mixt.data
## Tabulate 1st term of stochastic integral (inner convergence) in a real-life setting:
list.comp <- list(f1 = NULL, g1 = 'norm',
                  f2 = NULL, g2 = 'norm')
list.param <- list(f1 = NULL, g1 = list(mean = 2, sd = 0.7),
                   f2 = NULL, g2 = list(mean = 3, sd = 1.2))
U <- IBM_tabul_stochasticInteg(n.sim = 2, n.varCovMat = 20, sample1 = X.sim, sample2 = Y.sim,
                          min_size = NULL, comp.dist = list.comp, comp.param = list.param,
                                parallel = FALSE, n_cpu = 2)
```

---

`IBM_test_H0` *Equality test of unknown component distributions in two admixture models with IBM approach*

---

### Description

Two-sample test of the unknown component distribution in admixture models using Inversion - Best Matching (IBM) method. Recall that we have two admixture models with respective probability density functions (pdf) l1 = p1 f1 + (1-p1) g1 and l2 = p2 f2 + (1-p2) g2, where g1 and g2 are known pdf and l1 and l2 are observed. Perform the following hypothesis test: H0 : f1 = f2 versus H1 : f1 differs from f2.

### Usage

```
IBM_test_H0(
  sample1,
  sample2,
  known.p = NULL,
  comp.dist = NULL,
  comp.param = NULL,
  sim_U = NULL,
  n_sim_tab = 50,
```

```
  min_size = NULL,
  parallel = FALSE,
  n_cpu = 4
)
```

## Arguments

| | |
|---|---|
| sample1 | Observations of the first sample under study. |
| sample2 | Observations of the second sample under study. |
| known.p | (default to NULL) Numeric vector with two elements, the known (true) mixture weights. |
| comp.dist | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |
| comp.param | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)). |
| sim_U | Random draws of the inner convergence part of the contrast as defined in the IBM approach (see 'Details' below). |
| n_sim_tab | Number of simulated gaussian processes used in the tabulation of the inner convergence distribution in the IBM approach. |
| min_size | (default to NULL) In the k-sample case, useful to provide the minimal size among all samples. Otherwise, useless. |
| parallel | (default to FALSE) Boolean to indicate whether parallel computations are performed (speed-up the tabulation). |
| n_cpu | (default to 2) Number of cores used when parallelizing. |

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

A list of four elements, containing : 1) the test statistic value; 2) the rejection decision; 3) the p-value of the test, and 4) the estimated weights of the unknown component for each of the two admixture models.

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

### Examples

```
####### Under the null hypothesis H0 :
## Simulate data:
list.comp <- list(f1 = "norm", g1 = "norm",
                  f2 = "norm", g2 = "norm")
list.param <- list(f1 = list(mean = 1, sd = 1), g1 = list(mean = 2, sd = 0.7),
                   f2 = list(mean = 1, sd = 1), g2 = list(mean = 3, sd = 1.2))
X.sim <- rsimmix(n= 1100, unknownComp_weight=0.85, comp.dist = list(list.comp$f1,list.comp$g1),
                 comp.param = list(list.param$f1, list.param$g1))$mixt.data
Y.sim <- rsimmix(n= 1200, unknownComp_weight=0.75, comp.dist = list(list.comp$f2,list.comp$g2),
                 comp.param = list(list.param$f2, list.param$g2))$mixt.data
list.comp <- list(f1 = NULL, g1 = "norm",
                  f2 = NULL, g2 = "norm")
list.param <- list(f1 = NULL, g1 = list(mean = 2, sd = 0.7),
                   f2 = NULL, g2 = list(mean = 3, sd = 1.2))
IBM_test_H0(sample1=X.sim,sample2=Y.sim,known.p=NULL, comp.dist=list.comp,comp.param=list.param,
            sim_U = NULL, n_sim_tab = 6, min_size=NULL, parallel=FALSE, n_cpu=2)
```

---

## IBM_theoretical_contrast

*Theoretical contrast in the Inversion - Best Matching (IBM) method*

---

### Description

Defines the theoretical contrast in the IBM approach. Useful in case of simulation studies, since all parameters are known to the user. For further information about the considered contrast in the IBM approach, see 'Details' below.

### Usage

```
IBM_theoretical_contrast(
  par,
  theo.par,
  fixed.p.X = NULL,
  G = NULL,
  comp.dist,
  comp.param,
  sample1,
  sample2
)
```

## Arguments

| | |
|---|---|
| par | Numeric vector with two elements, corresponding to the two parameter values at which to compute the contrast. In practice the component weights for the two admixture models. |
| theo.par | Numeric vector with two elements, the known (true) mixture weights. |
| fixed.p.X | Arbitrary value chosen by the user for the component weight related to the unknown component of the first admixture model. Only useful for optimization when the known components of the two models are identical (G1=G2, leading to unidimensional optimization). |
| G | Distribution on which to integrate when calculating the contrast. |
| comp.dist | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. No unknown elements permitted. For instance, 'comp.dist' could be specified as follows: list(f1='rnorm', g1='norm', f2='rnorm', g2='norm'). |
| comp.param | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. No unknown elements permitted. For instance, 'comp.param' could be specified as follows: : list(f1 = list(mean=2,sd=0.3), g1 = list(mean=0,sd=1), f2 = list(mean=2,sd=0.3), g2 = list(mean=3,sd=1.1)). |
| sample1 | Observations of the first sample under study. |
| sample2 | Observations of the second sample under study. |

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

The theoretical contrast value evaluated at parameter values.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 1, sd = 0.1), g2 = list(mean = 5, sd = 2))
```

```
sample1 <- rsimmix(n=1500, unknownComp_weight=0.5, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=2000, unknownComp_weight=0.7, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Create the distribution on which the contrast will be integrated:
G <- stats::rnorm(n = 1000, mean = sample(c(sample1[['mixt.data']],sample2[['mixt.data']]),
                                          size = 1000, replace = TRUE),
                sd = stats::density(c(sample1[['mixt.data']],sample2[['mixt.data']]))$bw)
## Compute the theoretical contrast at parameters (p1,p2) = (0.2,0.7):
IBM_theoretical_contrast(par = c(0.2,0.7), theo.par = c(0.5,0.7), fixed.p.X = NULL, G = G,
                         comp.dist = list.comp, comp.param = list.param,
                      sample1 = sample1[['mixt.data']], sample2 = sample2[['mixt.data']])
```

---

`IBM_theoretical_gap` *Difference between unknown cumulative distribution functions of admixture models at some given point*

---

## Description

Compute the gap between the unknown cumulative distribution functions of the two considered admixture models at some given point, where each admixture model has probability distribution function (pdf) given by l where l = p*f + (1-p)*g. Uses the inversion method to do so, i.e. f = (1/p)(l - (1-p)g), where g represents the known component of the admixture model and p is the proportion of the unknown component. This difference must be integrated over some domain to compute the global discrepancy, as introduced in the paper presenting the IBM approach (see 'Details' below).

## Usage

```
IBM_theoretical_gap(z, par, known.p = c(0.5, 0.5), comp.dist, comp.param)
```

## Arguments

| | |
|---|---|
| `z` | Point at which the difference between the unknown component distributions of the two considered admixture models is computed. |
| `par` | Numeric vector with two elements, corresponding to the two parameter values at which to compute the gap. In practice the component weights for the two admixture models. |
| `known.p` | Numeric vector with two elements, the known (true) mixture weights. |
| `comp.dist` | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. No unknown elements permitted. For instance, 'comp.dist' could be specified as follows: list(f1='rnorm', g1='norm', f2='rnorm', g2='norm'). |

comp.param A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. No unknown elements permitted. For instance, 'comp.param' could be specified as follows: : list(f1 = list(mean=2,sd=0.3), g1 = list(mean=0,sd=1), f2 = list(mean=2,sd=0.3), g2 = list(mean=3,sd=1.1)).

## Details

See the paper presenting the IBM approach at the following HAL weblink: https://hal.archives-ouvertes.fr/hal-03201760

## Value

The gap between F1 and F2 (unknown components of the two admixture models), evaluated at the specified point.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 1, sd = 0.1), g2 = list(mean = 5, sd = 2))
IBM_theoretical_gap(z = 2.8, par = c(0.3,0.6), known.p = c(0.5,0.5),
                    comp.dist = list.comp, comp.param = list.param)
```

---

# is_equal_knownComp *Test for equality of the known components between two admixture models*

---

## Description

Test if the known components coming from the two two-components admixture models are the same.

## Usage

```
is_equal_knownComp(comp.dist, comp.param)
```

### Arguments

| | |
|---|---|
| comp.dist | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |
| comp.param | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)). |

### Value

A boolean (TRUE if the known components are the same, otherwise FALSE).

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

### Examples

```
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 2, sd = 0.3), g2 = list(mean = 0, sd = 1))
is_equal_knownComp(comp.dist = list.comp, comp.param = list.param)
```

---

## kernel_cdf *Kernel estimation*

---

### Description

Functions to perform the estimation of cumulative distribution function (cdf) by kernel estimators (with a non-gaussian kernel).

### Usage

```
kernel_cdf(u, h)
```

### Arguments

| | |
|---|---|
| u | the point at which the estimation is made. |
| h | window of the kernel estimation. |

### Value

the estimated value of the cdf.

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

### Examples

```
kernel_cdf(0.4,0.5)
```

---

## kernel_density *Kernel estimation*

---

### Description

Functions to perform the estimation of probability density function (pdf) by kernel estimators (with a non-gaussian kernel).

### Usage

```
kernel_density(u, h)
```

### Arguments

| | |
|---|---|
| u | the point at which the estimation is made. |
| h | window of the kernel estimation. |

### Value

the estimated value of the pdf.

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

### Examples

```
kernel_density(0.4,0.5)
```

knownComp_to_uniform *Transforms the known component of the admixture distribution to a Uniform distribution*

## Description

In admixture such that the probability density function (pdf) follows l = p*f + (1-p)*g, where p is the unknown weight and f is the unknown component distribution: transforms g of the two-component mixture ditribution to a Uniform distribution. Useful to use Patra and Sen estimator for the estimation of the unknown weight p.

## Usage

```
knownComp_to_uniform(data, comp.dist, comp.param)
```

## Arguments

| | |
|---|---|
| data | Observations of the sample under study, following an admixture distribution. |
| comp.dist | A list with two elements corresponding to component distributions (specified with R native names for these distributions) involved in the admixture model. Unknown elements must be specified as 'NULL' objects, e.g. when 'f' is unknown: list(f=NULL, g='norm'). |
| comp.param | A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Unknown elements must be specified as 'NULL' objects, e.g. if 'f' is unknown: list(f=NULL, g=list(mean=0,sd=1)). |

## Value

The transformed data, i.e. the transformed mixture distribution where the known component g now follows a Uniform(0,1) distribution.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5),
                   g1 = list(mean = 0, sd = 1))
sample1 <- rsimmix(n=1500, unknownComp_weight=0.5, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
plot_admix(sim.X = sample1[['mixt.data']], support = 'continuous')
## Transform the known component into a Uniform(0,1) distribution:
list.comp <- list(f1 = NULL, g1 = 'norm')
```

```
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1))
transformed_data <- knownComp_to_uniform(data = sample1[['mixt.data']],
                                         comp.dist = list.comp, comp.param = list.param)
plot_admix(sim.X = transformed_data, support = 'continuous')
```

---

`milkyWay` *Heliocentric velocity measured for the Milky Way (from Walker, M. G., M. Mateo, E. W. Olszewski, O. Y. Gnedin, X. Wang, B. Sen, and M. Woodroofe (2007). Velocity dispersion profiles of seven dwarf spheroidal galaxies. Astrophysical J. 667(1), L53–L56).*

---

## Description

Heliocentric velocity measured for the Milky Way (from Walker, M. G., M. Mateo, E. W. Olszewski, O. Y. Gnedin, X. Wang, B. Sen, and M. Woodroofe (2007). Velocity dispersion profiles of seven dwarf spheroidal galaxies. Astrophysical J. 667(1), L53–L56).

## Usage

```
milkyWay
```

## Format

A data frame with 170,601 rows and 1 column:

**V1** Heliocentric velocity measurements of the Milky way

## Source

https://www.aanda.org/articles/aa/full_html/2018/08/aa32905-18/aa32905-18.html

---

`orthoBasis_coef` *Compute expansion coefficients in a given orthonormal polynomial basis.*

---

## Description

Compute the coefficients corresponding to the decomposition of some density in a given orthonormal polynomial basis.

## Usage

```
orthoBasis_coef(
  data,
  comp.dist = NULL,
  comp.param = NULL,
  supp = c("Real", "Integer", "Positive", "Bounded.continuous"),
  degree,
  m = 3,
  other = NULL
)
```

## Arguments

| | |
|---|---|
| data | Observed sample from which the coefficients are calculated. Can be NULL if 'comp.dist' and 'comp.param' are specified. |
| comp.dist | (default to NULL) A list with two elements corresponding to component distributions (specified with R native names for these distributions) involved in the admixture model. Unknown elements must be specified as 'NULL' objects (for instance unknown 'f': list(f=NULL, g='norm')). |
| comp.param | (default to NULL) A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. Unknown elements must be specified as 'NULL' objects. For instance if 'f' is unknown: list(f = NULL, g = list(mean=0,sd=1)). |
| supp | Support of the density considered. |
| degree | Degree up to which the polynomial basis is built. |
| m | (default to 3) Only used when support is 'Integer'. Corresponds to the mean of the reference measure, i.e. Poisson(m). |
| other | (default to NULL) A list to precise bounds when the support is bounded, where the second and fourth elements give bounds. |

## Value

The list composed of 'degree' elements, each element being a numeric vector (with sample size) where each value represents the k-th order coefficient found when decomposing the density in the orthonormal polynomial basis.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
sample1 <- rnorm(n = 7000, mean = 3, sd = 1)
## Compute the expansion coefficients in the orthonormal polynomial basis:
coeff <- orthoBasis_coef(data = sample1, comp.dist = NULL, comp.param = NULL, supp = 'Real',
```

```
                          degree = 3, m = 3, other = NULL)
sapply(coeff, mean)
## No observed data and decomposition of the known component of the admixture model:
coeff <- orthoBasis_coef(data = NULL, comp.dist = list(NULL, 'norm'),
        comp.param=list(NULL,list(mean=3,sd=1)), supp = 'Real', degree=3, m=3, other = NULL)
sapply(coeff, mean)
```

---

`orthoBasis_test_H0` *Equality test of unknown components between two admixture models using polynomial basis expansions*

---

## Description

Test the null hypothesis (H0: f1=f2) using the decomposition of unknown densities of the two admixture distributions in an adequate orthonormal polynomial basis. Recall that we have two admixture models with respective probability density functions (pdf) l1 = p1*f1 + (1-p1)*g1 and l2* = *p2*f2 + (1-p2)*g2, where g1 and g2 are the only known elements. The admixture weights p1 and p2 thus have to be estimated. For further information on this method, see 'Details' below.

## Usage

```
orthoBasis_test_H0(
  data.X,
  data.Y,
  known.p = NULL,
  comp.dist = NULL,
  comp.param = NULL,
  known.coef = NULL,
  K = 3,
  nb.ssEch = 2,
  s = 0.49,
  var.explicit = FALSE,
  nb.echBoot = NULL,
 support = c("Real", "Integer", "Positive", "Bounded.continuous", "Bounded.discrete"),
  bounds.supp = NULL,
  est.method = c("BVdk", "PS"),
  uniformized.knownComp_data = NULL
)
```

## Arguments

| | |
|---|---|
| `data.X` | First observed sample following mixture distribution given by l1. |
| `data.Y` | Second observed sample following mixture distribution given by l2. |
| `known.p` | (default to NULL) Numeric vector with two elements, respectively the component weight for the unknown component in the first and in the second samples. |

`comp.dist` A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm').

`comp.param` A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)).

`known.coef` Coefficients in the polynomial basis expansion, corresponding to the known component densities g1 and g2.

`K` Number of coefficients considered for the polynomial basis expansion.

`nb.ssEch` Number of subsamples created from the original data to decorrelate the estimation of the different parameters.

`s` Rate at which the normalization factor is set in the penalization rule for model selection (in ]0,1/2[), see 'Details'.

`var.explicit` Boolean that allows to choose between explicit assessment of the variance of the test statistic or not (FALSE=bootstrap), FIXME : it seems that bootstrap procedure does not work in the context of admixtures.

`nb.echBoot` number of bootstrap samples if 'var.explicit' is set to FALSE.

`support` support of the densities under consideration, useful to choose the polynomial orthonormal basis.

`bounds.supp` (default to NULL) useful if support = 'bounded', a list of minimum and maximum bounds, specified as following: list( list(min.f1,min.g1,min.f2,min.g2) , list(max.f1,max.g1,max.f2,max.g2) )

`est.method` Estimation method to get the component weights, either 'PS' (Patra and Sen estimation) or 'BVdk' (Bordes and Vendekerkhove estimation).

`uniformized.knownComp_data` (default to NULL) Only useful if 'est.method' has been set to 'PS', and for real-life applications where the distribution of the known component of the admixture model is also unknown. In this case, this known component is previously made uniformly(0,1)-distributed by applying the empirical cumulative distribution of the known component function on the data. This means that all 'comp.dist' and 'comp.param' must be set to NULL.

## Details

See the paper on HAL website: https://hal.archives-ouvertes.fr/hal-02491127

## Value

A list with six elements containing: 1) the rejection decision; 2) the p-value of the test; 3) the test statistic; 4) the variance-covariance matrix of the test statistic; 5) selected rank for testing, and 6) estimates of the two component weights.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
###### Using Bordes and Vandekerkhove estimation (valid if symetric unknown component densities).
#### Under the null hypothesis H0.
## Simulate data:
list.comp <- list(f1 = "norm", g1 = "norm",
                  f2 = "norm", g2 = "norm")
list.param <- list(f1 = c(mean = 1, sd = 1), g1 = c(mean = 4, sd = 1),
                   f2 = c(mean = 1, sd = 1), g2 = c(mean = 5, sd = 0.5))
sim.X <- rsimmix(n = 250, unknownComp_weight=0.9, comp.dist = list(list.comp$f1,list.comp$g1),
                 comp.param = list(list.param$f1, list.param$g1))
sim.Y <- rsimmix(n = 300, unknownComp_weight=0.8, comp.dist = list(list.comp$f2,list.comp$g2),
                 comp.param = list(list.param$f2, list.param$g2))
plot_admix(sim.X = sim.X[['mixt.data']], sim.Y = sim.Y[['mixt.data']], support = "continuous")
## Perform the hypothesis test in real-life conditions:
list.comp <- list(f1 = NULL, g1 = "norm",
                  f2 = NULL, g2 = "norm")
list.param <- list(f1 = NULL, g1 = c(mean = 4, sd = 1),
                   f2 = NULL, g2 = c(mean = 5, sd = 0.5))
test <- orthoBasis_test_H0(data.X = sim.X[['mixt.data']], data.Y = sim.Y[['mixt.data']],
        known.p=NULL, comp.dist = list.comp, comp.param = list.param, known.coef=NULL, K=3,
           nb.ssEch = 2, s = 0.25, var.explicit=TRUE, nb.echBoot=NULL, support = 'Real',
            bounds.supp = NULL, est.method = 'BVdk', uniformized.knownComp_data = NULL)
test$decision
```

---

# PatraSen_cv_mixmodel *Cross-validation estimate (by Patra and Sen) of the unknown component weight as well as the unknown distribution in an admixture model*

---

## Description

Estimation of unknown elements (by Patra and Sen method) under the admixture model with probability density function l: l = p*f + (1-p)*g, where g is the known component of the two-component admixture, p is the unknown proportion of the unknown component distribution f. The estimated unknown component weight p is selected using a cross-validation technique that helps to choose the right penalization, see 'Details' below for further information.

## Usage

```
PatraSen_cv_mixmodel(
  data,
  folds = 10,
  reps = 1,
  cn.s = NULL,
  cn.length = NULL,
  gridsize = 200
)
```

## Arguments

| | |
|---|---|
| data | Sample where the known component density of the admixture model has been transformed into a Uniform(0,1) distribution. |
| folds | (default to 10) Number of folds used for cross-validation. |
| reps | (default to 1) Number of replications for cross-validation. |
| cn.s | (default to NULL) A sequence of 'c.n' to be used for cross-validation (vector of values). |
| cn.length | (default to NULL) Number of equally spaced tuning parameter (between .001 x log(log(n)) and 0.2 x log(log(n))). Values to search from. |
| gridsize | (default to 200) Number of equally spaced points (between 0 and 1) to evaluate the distance function. Larger values are more computationally intensive but also lead to more accurate estimates. |

## Details

See Patra, R.K. and Sen, B. (2016); Estimation of a Two-component Mixture Model with Applications to Multiple Testing; JRSS Series B, 78, pp. 869–893.

## Value

A list containing 'alp.hat' (estimate of the unknown component weight), 'Fs.hat' (list with elements 'x' and 'y' values for the function estimate of the unknown cumultaive distribution function), 'dist.out' which is an object of the class 'dist.fun' using the complete data.gen, 'c.n' the value of the tuning parameter used to compute the final estimate, and finally 'cv.out' which is an object of class 'cv.mixmodel'. The object is NULL if method is "fixed".

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
comp.dist <- list(f = 'norm', g = 'norm')
comp.param <- list(f = list(mean = 3, sd = 0.5),
                   g = list(mean = 0, sd = 1))
data1 <- rsimmix(n = 2000, unknownComp_weight = 0.3, comp.dist, comp.param)[['mixt.data']]
```

```
## Transform the known component of the admixture model into a Uniform(0,1) distribution:
comp.dist <- list(f = NULL, g = 'norm')
comp.param <- list(f = NULL, g = list(mean = 0, sd = 1))
data1_transfo <- knownComp_to_uniform(data = data1, comp.dist = list(comp.dist$f, comp.dist$g),
                                      comp.param = list(comp.param$f, comp.param$g))
plot(density(data1_transfo))
## Estimate the proportion of the unknown component of the admixture model:
PatraSen_cv_mixmodel(data = data1_transfo, folds = 3, reps = 1, cn.s = NULL,
                               cn.length = 3, gridsize = 100)$alp.hat
```

---

`PatraSen_density_est` *Compute the estimate of the density of the unknown component in an admixture model*

---

## Description

Compute by Patra and Sen technique the estimate of f.s (density corresponding to F.s) when f.s is known to be either decreasing or increasing.

## Usage

```
PatraSen_density_est(input, dec.density = TRUE)
```

## Arguments

| | |
|---|---|
| `input` | an R object of class 'cv.mixmodel' or 'mixmodel'. |
| `dec.density` | a boolean indicating whether the density is increasing or decreasing. |

## Details

See Patra, R.K. and Sen, B. (2016); Estimation of a Two-component Mixture Model with Applications to Multiple Testing; JRSS Series B, 78, pp. 869–893.

## Value

an estimator of the unknown component density.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
comp.dist <- list(f = 'norm', g = 'norm')
comp.param <- list(f = list(mean = 3, sd = 0.5), g = list(mean = 0, sd = 1))
data1 <- rsimmix(n = 2000, unknownComp_weight = 0.6, comp.dist, comp.param)[['mixt.data']]
data1_transfo <- knownComp_to_uniform(data = data1, comp.dist = list(comp.dist$f, comp.dist$g),
                                      comp.param = list(comp.param$f, comp.param$g))
res <- PatraSen_cv_mixmodel(data = data1_transfo, folds = 3, reps = 1, cn.s = NULL,
                            cn.length = 3, gridsize = 200)
PatraSen_density_est(res, dec.density = TRUE)
```

---

# PatraSen_dist_calc *Compute the distance to be minimized using Patra and Sen estimation technique in admixture models*

---

## Description

Compute the distance to be minimized using Patra and Sen estimation technique by integrating along some given grid the appropriate distance. For further developments, see 'Details' below.

## Usage

```
PatraSen_dist_calc(data, gridsize = 200)
```

## Arguments

| | |
|---|---|
| `data` | Sample where the known component density of the admixture model has been transformed into a Uniform(0,1) distribution. |
| `gridsize` | Gridsize to make the computations. |

## Details

See Patra, R.K. and Sen, B. (2016); Estimation of a Two-component Mixture Model with Applications to Multiple Testing; JRSS Series B, 78, pp. 869–893.

## Value

a list containing the evaluated distance and some additional information.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
comp.dist <- list(f = 'norm', g = 'norm')
comp.param <- list(f = list(mean = 3, sd = 0.5), g = list(mean = 0, sd = 1))
data1 <- rsimmix(n = 3000, unknownComp_weight = 0.6, comp.dist, comp.param)[['mixt.data']]
data1_transfo <- knownComp_to_uniform(data = data1, comp.dist = list(comp.dist$f, comp.dist$g),
                                      comp.param = list(comp.param$f, comp.param$g))
PatraSen_dist_calc(data = data1_transfo, gridsize = 200)
```

---

# PatraSen_est_mix_model

*Estimate by Patra and Sen the unknown component weight as well as the unknown distribution in admixture models*

---

## Description

Estimation of unknown elements (by Patra and Sen method) under the admixture model with probability density function l: l = p*f + (1-p)*g, where g is the known component of the two-component mixture, p is the unknown proportion of the unknown component distribution f. More information in 'Details' below concerning the estimation method.

## Usage

```
PatraSen_est_mix_model(
  data,
  method = c("lwr.bnd", "fixed", "cv"),
  c.n = NULL,
  folds = 10,
  reps = 1,
  cn.s = NULL,
  cn.length = 100,
  gridsize = 600
)
```

## Arguments

| | |
|---|---|
| data | Sample where the known component density of the admixture model has been transformed into a Uniform(0,1) distribution. |
| method | Either 'fixed' or 'cv', depending on whether compute the estimate based on the value of 'c.n' or use cross-validation for choosing 'c.n' (tuning parameter). |
| c.n | A positive number, with default value equal to 0.1 log(log(n)), where 'n' is the length of the observed sample. |
| folds | Number of folds used for cross-validation, default is 10. |
| reps | Number of replications for cross-validation, default is 1. |

| | |
|---|---|
| cn.s | A sequence of 'c.n' to be used for cross-validation (vector of values). Default is equally spaced grid of 100 values between .001 x log(log(n)) and 0.2 x log(log(n)). |
| cn.length | (default to 100) Number of equally spaced tuning parameter (between .001 x log(log(n)) and 0.2 x log(log(n))). Values to search from. |
| gridsize | (default to 600) Number of equally spaced points (between 0 and 1) to evaluate the distance function. Larger values are more computationally intensive but also lead to more accurate estimates. |

## Details

See Patra, R.K. and Sen, B. (2016); Estimation of a Two-component Mixture Model with Applications to Multiple Testing; JRSS Series B, 78, pp. 869–893.

## Value

A list containing 'alp.hat' (estimate of the unknown component weight), 'Fs.hat' (list with elements 'x' and 'y' values for the function estimate of the unknown cumultaive distribution function), 'dist.out' which is an object of the class 'dist.fun' using the complete data.gen, 'c.n' the value of the tuning parameter used to compute the final estimate, and finally 'cv.out' which is an object of class 'cv.mixmodel'. The object is NULL if method is "fixed".

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
## Simulate data:
list.comp <- list(f = 'norm', g = 'norm')
list.param <- list(f = list(mean = 3, sd = 0.5),
                   g = list(mean = 0, sd = 1))
data1 <- rsimmix(n = 1500, unknownComp_weight = 0.8, list.comp, list.param)[['mixt.data']]
## Transform the known component of the admixture model into a Uniform(0,1) distribution:
list.comp <- list(f = NULL, g = 'norm')
list.param <- list(f = NULL, g = list(mean = 0, sd = 1))
data1_transfo <- knownComp_to_uniform(data = data1, comp.dist=list.comp, comp.param=list.param)
PatraSen_est_mix_model(data = data1_transfo, method = 'fixed',
                       c.n = 0.1*log(log(length(data1_transfo))), gridsize = 1000)$alp.hat
```

---

# plot_admix *Plot the density of some given sample(s)*

---

## Description

Plot the density of the sample(s) with optional arguments to improve the visualization.

### Usage

```
plot_admix(
  sim.X,
  sim.Y = NULL,
  user.bounds = NULL,
  support = c("continuous", "discrete"),
  case = ""
)
```

### Arguments

| | |
|---|---|
| sim.X | First sample from which the density will be plotted. |
| sim.Y | (default to NULL) Second sample from which the density will be plotted. |
| user.bounds | (default to NULL) Bounds to limit the range of x-axis when plotting. |
| support | Support of the distributions, to know whether density plot or histogram should be displayed. |
| case | Used for titles. |

### Value

a plot with the densities of the samples provided as inputs.

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

### Examples

```
comp.dist <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
comp.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = -2, sd = 0.8), g2 = list(mean = 2, sd = 0.9))
sim.X <- rsimmix(n=2000, unknownComp_weight = 0.7, comp.dist = list(comp.dist$f1,comp.dist$g1),
                 comp.param = list(comp.param$f1,comp.param$g1))
sim.Y <- rsimmix(n=2000, unknownComp_weight = 0.4, comp.dist = list(comp.dist$f2,comp.dist$g2),
                 comp.param = list(comp.param$f2,comp.param$g2))
plot_admix(sim.X[['mixt.data']], sim.Y[['mixt.data']],
           user.bounds = c(-6,6), support = 'continuous')
```

---

## plot_decontamin_cdf *Plot the decontaminated cumulative distribution function (CDF) of the unknown component for an estimated admixture model*

---

## Description

Plot the decontaminated CDF of the unknown component of the admixture model under study, after inversion of the admixture CDF. Recall that an admixture model follows the cumulative distribution function (CDF) L, where L = p*F + (1-p)*G, with g a known CDF and p and f unknown quantities.

## Usage

```
plot_decontamin_cdf(x_val, decontamin_cdf, add_plot = FALSE)
```

## Arguments

| | |
|---|---|
| x_val | A vector of x-axis values at which to plot the decontaminated cumulative distribution function F. |
| decontamin_cdf | An object from function 'decontamin_cdf_unknownComp', containing the unknown decontaminated distribution function, as well as the support of the distribution (either discrete or continuous). |
| add_plot | A boolean (TRUE by default) specifying if one plots the decontaminated density over an existing plot. Used for visual comparison purpose. |

## Details

The decontaminated CDF is obtained by inverting the admixture CDF, given by L = p*F + (1-p)*G, to isolate the unknown component F after having estimated p and L.

## Value

The plot of the decontaminated cumulative distribution function.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
####### Continuous support:
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 3, sd = 0.5), g2 = list(mean = 5, sd = 2))
sample1 <- rsimmix(n=3000, unknownComp_weight=0.7, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=2500, unknownComp_weight=0.8, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'norm',
                  f2 = NULL, g2 = 'norm')
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1),
                   f2 = NULL, g2 = list(mean = 5, sd = 2))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
```

```
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
## Determine the decontaminated version of the unknown CDF by inversion:
res1 <- decontamin_cdf_unknownComp(sample1 = sample1[['mixt.data']],
                           comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                           estim.p = estimate$prop.estim[1])
res2 <- decontamin_cdf_unknownComp(sample1 = sample2[['mixt.data']],
                           comp.dist = list.comp[3:4], comp.param = list.param[3:4],
                           estim.p = estimate$prop.estim[2])
plot_decontamin_cdf(x_val = seq(from=-1, to=5, length.out=30), decontamin_cdf = res1,
                    add_plot = FALSE)
plot_decontamin_cdf(x_val = seq(from=-1, to=5, length.out=30), decontamin_cdf = res2,
                    add_plot = TRUE)
####### Countable discrete support:
list.comp <- list(f1 = 'pois', g1 = 'pois',
                  f2 = 'pois', g2 = 'pois')
list.param <- list(f1 = list(lambda = 3), g1 = list(lambda = 2),
                   f2 = list(lambda = 3), g2 = list(lambda = 4))
sample1 <- rsimmix(n=4000, unknownComp_weight=0.7, comp.dist = list(list.comp$f1,list.comp$g1),
                                           comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=3000, unknownComp_weight=0.9, comp.dist = list(list.comp$f2,list.comp$g2),
                                           comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'pois',
                  f2 = NULL, g2 = 'pois')
list.param <- list(f1 = NULL, g1 = list(lambda = 2),
                   f2 = NULL, g2 = list(lambda = 4))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
res1 <- decontamin_cdf_unknownComp(sample1 = sample1[['mixt.data']],
                           comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                           estim.p = estimate$prop.estim[1])
res2 <- decontamin_cdf_unknownComp(sample1 = sample2[['mixt.data']],
                           comp.dist = list.comp[3:4], comp.param = list.param[3:4],
                           estim.p = estimate$prop.estim[2])
plot_decontamin_cdf(x_val = seq(from=0, to=15, by=1), decontamin_cdf = res1,
                    add_plot = FALSE)
plot_decontamin_cdf(x_val = seq(from=0, to=15, by=1), decontamin_cdf = res2,
                    add_plot = TRUE)
####### Finite discrete support:
list.comp <- list(f1 = 'multinom', g1 = 'multinom',
                  f2 = 'multinom', g2 = 'multinom')
list.param <- list(f1 = list(size=1, prob=c(0.3,0.4,0.3)), g1 = list(size=1, prob=c(0.6,0.3,0.1)),
             f2 = list(size=1, prob=c(0.3,0.4,0.3)), g2 = list(size=1, prob=c(0.2,0.6,0.2)))
sample1 <- rsimmix(n=5000, unknownComp_weight=0.7, comp.dist = list(list.comp$f1,list.comp$g1),
                                           comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=4000, unknownComp_weight=0.9, comp.dist = list(list.comp$f2,list.comp$g2),
                                           comp.param=list(list.param$f2,list.param$g2))
list.comp <- list(f1 = NULL, g1 = 'multinom',
                  f2 = NULL, g2 = 'multinom')
list.param <- list(f1 = NULL, g1 = list(size=1, prob=c(0.6,0.3,0.1)),
                   f2 = NULL, g2 = list(size=1, prob=c(0.2,0.6,0.2)))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
```

```
res1 <- decontamin_cdf_unknownComp(sample1 = sample1[['mixt.data']],
                           comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                           estim.p = estimate$prop.estim[1])
res2 <- decontamin_cdf_unknownComp(sample1 = sample2[['mixt.data']],
                           comp.dist = list.comp[3:4], comp.param = list.param[3:4],
                           estim.p = estimate$prop.estim[2])
plot_decontamin_cdf(x_val = seq(from=0, to=5, by=1), decontamin_cdf = res1,
                    add_plot = FALSE)
plot_decontamin_cdf(x_val = seq(from=0, to=5, by=1), decontamin_cdf = res2,
                    add_plot = TRUE)
```

---

`plot_decontamin_density` *Plot the decontaminated density of the unknown component for an estimated admixture model*

---

## Description

Plot the decontaminated density of the unknown component in the admixture model under study, after inversion of the admixture cumulative distribution function. Recall that an admixture model follows the cumulative distribution function (CDF) L, where L = p*F + (1-p)*G, with g a known CDF and p and f unknown quantities.

## Usage

```
plot_decontamin_density(x_val, decontamin_obj, add_plot = FALSE)
```

## Arguments

| | |
|---|---|
| `x_val` | A vector of X-axis values at which to plot the decontaminated density f. |
| `decontamin_obj` | An object from function 'decontamin_density_unknownComp', containing the X-axis values and range for plotting, the corresponding Y-axis values, and the support over which to plot the results. |
| `add_plot` | (default to FALSE) A boolean specifying if one plots the decontaminated density over an existing plot. Used for visual comparison purpose. |

## Details

The decontaminated density is obtained by inverting the admixture density, given by l = p*f + (1-p)*g, to isolate the unknown component f after having estimated p.

## Value

The plot of the decontaminated density.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
####### Continuous support:
## Simulate data:
list.comp <- list(f1 = 'norm', g1 = 'norm',
                  f2 = 'norm', g2 = 'norm')
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 3, sd = 0.5), g2 = list(mean = 5, sd = 2))
sample1 <- rsimmix(n=3000, unknownComp_weight=0.7, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=2500, unknownComp_weight=0.8, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'norm',
                  f2 = NULL, g2 = 'norm')
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1),
                   f2 = NULL, g2 = list(mean = 5, sd = 2))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
## Determine the decontaminated version of the unknown density by inversion:
res1 <- decontamin_density_unknownComp(sample1 = sample1[['mixt.data']],
                                comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                                      estim.p = estimate$prop.estim[1])
res2 <- decontamin_density_unknownComp(sample1 = sample2[['mixt.data']],
                                comp.dist = list.comp[3:4], comp.param = list.param[3:4],
                                       estim.p = estimate$prop.estim[2])
## Use appropriate sequence of x values:
plot_decontamin_density(x_val = seq(from=0, to=6, length.out=100), decontamin_obj = res1,
                        add_plot = FALSE)
plot_decontamin_density(x_val = seq(from=0, to=6, length.out=100), decontamin_obj = res2,
                        add_plot = TRUE)
####### Countable discrete support:
list.comp <- list(f1 = 'pois', g1 = 'pois',
                  f2 = 'pois', g2 = 'pois')
list.param <- list(f1 = list(lambda = 3), g1 = list(lambda = 2),
                   f2 = list(lambda = 3), g2 = list(lambda = 4))
sample1 <- rsimmix(n=4000, unknownComp_weight=0.7, comp.dist = list(list.comp$f1,list.comp$g1),
                                          comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=3500, unknownComp_weight=0.85, comp.dist = list(list.comp$f2,list.comp$g2),
                                          comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'pois',
                  f2 = NULL, g2 = 'pois')
list.param <- list(f1 = NULL, g1 = list(lambda = 2),
                   f2 = NULL, g2 = list(lambda = 4))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
## Determine the decontaminated version of the unknown density by inversion:
res1 <- decontamin_density_unknownComp(sample1 = sample1[['mixt.data']],
                                comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                                      estim.p = estimate$prop.estim[1])
res2 <- decontamin_density_unknownComp(sample1 = sample2[['mixt.data']],
                                comp.dist = list.comp[3:4], comp.param = list.param[3:4],
```

```
                                        estim.p = estimate$prop.estim[2])
## Use appropriate sequence of x values:
plot_decontamin_density(x_val = seq(from=0, to=15, by=1), decontamin_obj = res1,
                        add_plot = FALSE)
plot_decontamin_density(x_val = seq(from=0, to=15, by=1), decontamin_obj = res2,
                        add_plot = TRUE)
####### Finite discrete support:
list.comp <- list(f1 = 'multinom', g1 = 'multinom',
                  f2 = 'multinom', g2 = 'multinom')
list.param <- list(f1 = list(size=1, prob=c(0.3,0.4,0.3)), g1 = list(size=1, prob=c(0.6,0.3,0.1)),
             f2 = list(size=1, prob=c(0.3,0.4,0.3)), g2 = list(size=1, prob=c(0.2,0.6,0.2)))
sample1 <- rsimmix(n=4000, unknownComp_weight=0.8, comp.dist = list(list.comp$f1,list.comp$g1),
                                        comp.param=list(list.param$f1,list.param$g1))
sample2 <- rsimmix(n=3500, unknownComp_weight=0.9, comp.dist = list(list.comp$f2,list.comp$g2),
                                        comp.param=list(list.param$f2,list.param$g2))
## Estimate the mixture weight in each of the sample in real-life setting:
list.comp <- list(f1 = NULL, g1 = 'multinom',
                  f2 = NULL, g2 = 'multinom')
list.param <- list(f1 = NULL, g1 = list(size=1, prob=c(0.6,0.3,0.1)),
                   f2 = NULL, g2 = list(size=1, prob=c(0.2,0.6,0.2)))
estimate <- IBM_estimProp(sample1[['mixt.data']], sample2[['mixt.data']], comp.dist = list.comp,
                        comp.param = list.param, with.correction = FALSE, n.integ = 1000)
## Determine the decontaminated version of the unknown density by inversion:
res1 <- decontamin_density_unknownComp(sample1 = sample1[['mixt.data']],
                                comp.dist = list.comp[1:2], comp.param = list.param[1:2],
                                      estim.p = estimate$prop.estim[1])
res2 <- decontamin_density_unknownComp(sample1 = sample2[['mixt.data']],
                                comp.dist = list.comp[3:4], comp.param = list.param[3:4],
                                       estim.p = estimate$prop.estim[2])
## Use appropriate sequence of x values:
plot_decontamin_density(x_val = seq(from=0, to=6, by=1), decontamin_obj = res1,
                        add_plot = FALSE)
plot_decontamin_density(x_val = seq(from=0, to=6, by=1), decontamin_obj = res2,
                        add_plot = TRUE)
```

---

`poly_orthonormal_basis` *Build an orthonormal basis to decompose some given probability density function*

---

## Description

Build an orthonormal basis, needed to decompose the probability density function (pdf) of the unknown component from the admixture, depending on the support under consideration.

## Usage

```
poly_orthonormal_basis(
 support = c("Real", "Integer", "Positive", "Bounded.continuous", "Bounded.discrete"),
```

```
  deg,
  x,
  m
)
```

## Arguments

| | |
|---|---|
| support | Support of the random variables implied in the two-component mixture distribution. |
| deg | Degree up to which the basis is built. |
| x | (NULL by default) Only used when support is 'Integer'. The point at which the polynomial value will be evaluated. |
| m | (NULL by default) Only used when support is 'Integer'. Corresponds to the mean of the reference measure, i.e. Poisson(m). |

## Value

the orthonormal polynomial basis used to decompose the density of the unknown component of the mixture distribution.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
poly_orthonormal_basis(support = 'Real', deg = 10, x = NULL, m = NULL)
```

---

# rsimmix *Simulation of a two-component mixture model*

---

## Description

Simulate a two-component mixture model following the probability density function (pdf) l such that l = p*f + (1-p)*g, with f and g mixture component distributions, and p the mixture weight.

## Usage

```
rsimmix(
  n = 1000,
  unknownComp_weight = 0.5,
  comp.dist = list(f = "norm", g = "norm"),
  comp.param = list(f = c(mean = 0, sd = 1), g = c(mean = 2, sd = 1))
)
```

## Arguments

| | |
|---|---|
| n | Number of observations to be drawn. |
| unknownComp_weight | Weight of the component distribution f (representing the unknown component in admixture models). |
| comp.dist | A list with two elements corresponding to the component distributions (specified with R native names for these distributions) involved in the mixture model. These elements respectively refer to the two components f and g. No unknown elements permitted. For instance, 'comp.dist' could be set equal to list(f = 'rnorm', g = 'norm'). |
| comp.param | A list with two elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. These elements respectively refer to the parameters of f and g distributions of the mixture model. No unknown elements permitted. For instance, 'comp.param' could be set equal to list(f=list(mean=2,sd=0.3), g=list(mean=0,sd=1)). |

## Value

A list of three components. The first, named 'mixt.data', is the simulated sample from the specified mixture distribution. The second, named 'unknown.data', refers to the data simulated corresponding to the distribution f. The third, named 'known.data', corresponds to the observations affiliated to the known component g.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
sim.X <- rsimmix(n = 2000, unknownComp_weight = 0.7, comp.dist = list(f = 'norm', g = 'norm'),
              comp.param = list(f = list(mean = 3, sd = 0.5), g = list(mean = 0, sd = 1)))
class(sim.X)
attributes(sim.X)
plot_admix(sim.X = sim.X$mixt.data, sim.Y = NULL, user.bounds = NULL, support = 'continuous')
```

---

# rsimmix_mix *Simulation of a two-component mixture with one component following a two-component mixture*

---

## Description

simulate a two-component admixture model, where the first component is a mixture itself

## Usage

```
rsimmix_mix(n, m, s, p, a)
```

### Arguments

| | |
|---|---|
| n | is the number of observations to be drawn |
| m | the mean (up to the shift a) of the unknown components |
| s | the standard deviation of the unknown components |
| p | the weight of the unknown component (itself a mixture). |
| a | the shift of the mean for the two distributions that are embedded in the unknown component |

### Value

a list containing the data generated from a mixture of mixture distribution, the data where the known component density has been made uniform(0,1), and the known data (corresponding to the part of data generated from the known component density).

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

### Examples

```
sample1 <- rsimmix_mix(n = 3000, m = 5, s = 0.5, p = 0.3, a = 2)[['mixt.data']]
plot(stats::density(sample1))
```

---

## silhouette_criterion *Compute the silhouette criterion related to the K populations that were clustered*

---

### Description

Compute the silhouette criterion in k-sample clustering of admixture models.

### Usage

```
silhouette_criterion(clusters_obj)
```

### Arguments

| | |
|---|---|
| clusters_obj | an object obtained from function 'k_samples_clustering'. |

### Value

the silhouette criterion computed for each of the K populations under study.

### Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
###### Case study with 5 populations to cluster on R+ with Gamma-Exponential mixtures.
## Simulate data (chosen parameters indicate 3 clusters (populations (1,3), (2,5) and 4)!):
list.comp <- list(f1 = "gamma", g1 = "exp",
                  f2 = "gamma", g2 = "exp",
                  f3 = "gamma", g3 = "gamma",
                  f4 = "exp", g4 = "exp",
                  f5 = "gamma", g5 = "exp")
list.param <- list(f1 = list(shape = 16, rate = 4), g1 = list(rate = 1/3.5),
                   f2 = list(shape = 14, rate = 2), g2 = list(rate = 1/5),
                   f3 = list(shape = 16, rate = 4), g3 = list(shape = 12, rate = 2),
                   f4 = list(rate = 1/2), g4 = list(rate = 1/7),
                   f5 = list(shape = 14, rate = 2), g5 = list(rate = 1/6))
A.sim <- rsimmix(n=8000, unknownComp_weight=0.7, comp.dist = list(list.comp$f1,list.comp$g1),
                 comp.param = list(list.param$f1, list.param$g1))$mixt.data
B.sim <- rsimmix(n=8000, unknownComp_weight=0.6, comp.dist = list(list.comp$f2,list.comp$g2),
                 comp.param = list(list.param$f2, list.param$g2))$mixt.data
C.sim <- rsimmix(n=8000, unknownComp_weight=0.5, comp.dist = list(list.comp$f3,list.comp$g3),
                 comp.param = list(list.param$f3, list.param$g3))$mixt.data
D.sim <- rsimmix(n=8000, unknownComp_weight=0.4, comp.dist = list(list.comp$f4,list.comp$g4),
                 comp.param = list(list.param$f4, list.param$g4))$mixt.data
E.sim <- rsimmix(n=8000, unknownComp_weight=0.3, comp.dist = list(list.comp$f5,list.comp$g5),
                 comp.param = list(list.param$f5, list.param$g5))$mixt.data
## Look for the clusters:
list.comp <- list(f1 = NULL, g1 = "exp",
                  f2 = NULL, g2 = "exp",
                  f3 = NULL, g3 = "gamma",
                  f4 = NULL, g4 = "exp",
                  f5 = NULL, g5 = "exp")
list.param <- list(f1 = NULL, g1 = list(rate = 1/3.5),
                   f2 = NULL, g2 = list(rate = 1/5),
                   f3 = NULL, g3 = list(shape = 12, rate = 2),
                   f4 = NULL, g4 = list(rate = 1/7),
                   f5 = NULL, g5 = list(rate = 1/6))
clusters <- admix_clustering(samples = list(A.sim,B.sim,C.sim,D.sim,E.sim), n_sim_tab = 8,
              comp.dist = list.comp, comp.param = list.param, parallel = FALSE, n_cpu = 2)
clusters
silhouette_criterion(clusters_obj = clusters)
```

---

`sim_gaussianProcess` *Simulation of a Gaussian process*

---

## Description

Simulate the trajectory of a Gaussian process, given a mean vector and a variance-covariance structure.

## Usage

```
sim_gaussianProcess(
  mean_vec,
  varCov_mat,
  from = 0,
  to = 1,
  start = 0,
  nb.points = 10
)
```

## Arguments

| | |
|---|---|
| mean_vec | Vector (if multimensional) of means for the increments following gaussian distribution. |
| varCov_mat | Corresponding variance-covariance structure. |
| from | Initial time point at which the process is simulated. |
| to | Last time point at which the process is simulated. |
| start | Useful if the user wants to make the trajectory start from some given value. |
| nb.points | Number of points at which the process is simulated. |

## Value

The trajectory of the Gaussian processes after simulating the multivariate Gaussian distributions with specified variance-covariance structure.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
list.comp <- list(f1 = "norm", g1 = "norm")
list.param <- list(f1 = list(mean = 12, sd = 0.4),
                   g1 = list(mean = 16, sd = 0.7))
sample1 <- rsimmix(n = 2000, unknownComp_weight = 0.5, comp.dist = list.comp,
                   comp.param = list.param)$mixt.data
## First get the variance-covariance matrix of the empirical process (Donsker correlation):
cov_mat <- .Call('_admix_estimVarCov_empProcess_Rcpp', PACKAGE = 'admix',
                 seq(from = min(sample1), to = max(sample1), length.out = 100), sample1)
## Plug it into the simulation of the gaussian process:
B1 <- sim_gaussianProcess(mean_vec=rep(0,nrow(cov_mat)), varCov_mat=cov_mat, from=min(sample1),
                          to = max(sample1), start = 0, nb.points = nrow(cov_mat))
plot(x = B1$dates, y = B1$traj1, type="l", xlim = c(min(sample1),max(sample1)), ylim = c(-1,1))
```

`stmf_small` *Short-term Mortality Fluctuations (STMF) data series, restricted to 6 countries (Belgium, France, Italy, Netherlands, Spain, Germany).*

## Description

Weekly death counts provide the most objective and comparable way of assessing the scale of short-term mortality elevations across countries (32 countries) and time. Extraction date: 09/21/2020.

## Usage

```
stmf_small
```

## Format

A data frame with 88146 rows and 19 variables:

**CountryCode** Mortality database country code

**Year** Year

**Week** Week number

**Sex** Gender ('m': male, 'f': female, 'b': both)

**D0_14** Age range 0-14

**D15_64** Age range 15-64

**D65_74** Age range 65-74

**D75_84** Age range 75-84

**D85p** Age range 85-+

**DTotal** Count of deaths for all ages combined

**R0_14** Crude death rate for age range 0-14

**R15_64** Crude death rate for age range 15-64

**R65_74** Crude death rate for age range 65-74

**R75_84** Crude death rate for age range 75-84

**R85p** Crude death rate for age range 85-+

**RTotal** Crude death rate for all ages combined

**Split** Indicates if data were split from aggregated age groups (0 if the original data has necessary detailed age scale). For example, if the original age scale was 0-4, 5-29, 30-65, 65+, then split will be equal to 1

**SplitSex** Indicates if the original data are available by sex (0) or data are interpolated (1)

**Forecast** Equals 1 for all years where forecasted population exposures were used to calculate weekly death rates

## Source

https://www.mortality.org

two_samples_test *Two-samples hypothesis test on the unknown component in admixture models*

## Description

Test hypothesis on the unknown component of admixture models using different estimation techniques, and different testing strategies.

## Usage

```
two_samples_test(
  sample1,
  sample2,
  known.p = NULL,
  comp.dist = NULL,
  comp.param = NULL,
  method = c("ICV", "Poly"),
  n_sim_tab = NULL,
  K = 3,
  support = c("Real", "Positive", "Integer", "Bounded.continuous"),
  est.method = c("BVdk", "PS"),
  s = 0.49,
  nb.ssEch = 2,
  var.explicit = F,
  nb.echBoot = NULL,
  bounds.supp = NULL,
  parallel = FALSE,
  n_cpu = 2
)
```

## Arguments

| | |
|---|---|
| sample1 | First observed sample with mixture distribution given by l1 = p1*f1 + (1-p1)*g1, where f1 and p1 are unknown and g1 is known. |
| sample2 | Second observed sample with mixture distribution given by l2 = p2*f2 + (1-p2)*g2, where f2 and p2 are unknown and g2 is known. |
| known.p | (default to NULL) The true component weights p1 and p2 if known, only useful in simulation studies. |
| comp.dist | A list with four elements corresponding to the component distributions (specified with R native names for these distributions) involved in the two admixture models. The two first elements refer to the unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.dist' could be specified as follows: list(f1=NULL, g1='norm', f2=NULL, g2='norm'). |

| | |
|---|---|
| comp.param | A list with four elements corresponding to the parameters of the component distributions, each element being a list itself. The names used in this list must correspond to the native R argument names for these distributions. The two first elements refer to the parameters of unknown and known components of the 1st admixture model, and the last two ones to those of the second admixture model. If there are unknown elements, they must be specified as 'NULL' objects. For instance, 'comp.param' could be specified as follows: : list(f1=NULL, g1=list(mean=0,sd=1), f2=NULL, g2=list(mean=3,sd=1.1)). |
| method | Method used for testing. Choose either 'Poly' or 'ICV'. 'Poly' refers to comparison of expansion coefficients in polynomial orthonormal basis, whereas 'ICV' refers to the Inner Convergence property obtained when using the IBM approach. More details are provided below in 'Details'. |
| n_sim_tab | (Only with 'ICV' method) Number of simulated gaussian processes used for the tabulation of the Inner Convergence distribution in IBM approach. |
| K | (Only for 'Poly' method) Number of coefficients considered for the polynomial basis expansion. |
| support | (Only for 'Poly' method) Support of the densities under consideration, useful to choose the polynomial orthonormal basis. One of 'Real', 'Integer', 'Positive', or 'Bounded.continuous'. |
| est.method | (Only for 'Poly' method) Either 'BVdk' (Bordes and Valdekerkhove estimation technique) or 'PS' (Patra and Sen estimation technique). The latter should not be used since the estimators plugged into the test statistic are not square-root n consistent. More details are given in Section 'Details' below. |
| s | (Only for 'Poly' method) Rate at which the normalization factor is set in the penalization rule for model selection (in ]0,1/2[). |
| nb.ssEch | (Only with 'Poly' method) Number of subsamples created from original data to decorrelate the estimation of the parameters. |
| var.explicit | (Only with 'Poly' method) Boolean that enables to choose between explicit evaluation of the variance of the test statistic or not (FALSE=bootstrap). FIXME: it seems that bootstrap procedure does not work in the context of admixtures. |
| nb.echBoot | (Only with 'Poly' method) Number of bootstrap samples if 'var.explicit' is set to FALSE. |
| bounds.supp | (Only with 'Poly' method) default to NULL. Useful if support = 'bounded.continuous', a list of minimum and maximum bounds, specified as follows: list( list(min.f1,min.g1,min.f2,min.g2) , list(max.f1,max.g1,max.f2,max.g2) ) |
| parallel | Boolean to indicate whether parallel computations are performed (speed-up the tabulation). |
| n_cpu | Number of cores used when parallelizing. |

## Details

Here as some details concerning the different methods that can be choosen: i) 'Poly' relies on two-sample testing strategy where each unknown component density is decomposed in an orthonormal polynomial basis, and the estimation of the component weights related to the two two-component admixture models can be performed either using Patra and Sen estimator (despite the latter is not

square-root n consistent and thus should not be used in such hypothesis tests), or by Bordes and Vandekerkhove estimation technique (if the unknown component density is symmetric); ii) 'ICV' refers to Inversion - Best Matching strategy which has no constraints except that we need to handle two samples.

## Value

The decision of the test with further information such as p-value and others, depending on the method used.

## Author(s)

Xavier Milhaud xavier.milhaud.research@gmail.com

## Examples

```
##### Under the null hypothesis H0 :
## Simulate data:
list.comp <- list(f1 = "norm", g1 = "norm",
                  f2 = "norm", g2 = "norm")
list.param <- list(f1 = list(mean = 3, sd = 0.5), g1 = list(mean = 0, sd = 1),
                   f2 = list(mean = 3, sd = 0.5), g2 = list(mean = 6, sd = 1.2))
sample1 <- rsimmix(n=250, unknownComp_weight=0.85, comp.dist = list(list.comp$f1,list.comp$g1),
                   comp.param = list(list.param$f1,list.param$g1))[['mixt.data']]
sample2 <- rsimmix(n=300, unknownComp_weight=0.8, comp.dist = list(list.comp$f2,list.comp$g2),
                   comp.param = list(list.param$f2,list.param$g2))[['mixt.data']]
plot_admix(sample1, sample2, NULL, support='continuous')
##### Performs the test by the different methods :
list.comp <- list(f1 = NULL, g1 = "norm",
                  f2 = NULL, g2 = "norm")
list.param <- list(f1 = NULL, g1 = list(mean = 0, sd = 1),
                   f2 = NULL, g2 = list(mean = 6, sd = 1.2))
## Using expansion coefficients in orthonormal polynomial basis:
two_samples_test(sample1=sample1, sample2=sample2, comp.dist=list.comp, comp.param=list.param,
                 method = 'Poly', K = 3, support = 'Real', est.method = 'BVdk', s = 0.4,
                 nb.ssEch = 2, var.explicit = TRUE)
```

# Index