Description  Given a partition resulting from any clustering algorithm, the implemented tests al-
low valid post-clustering inference by testing if a given variable significantly sepa-
rates two of the estimated clusters.
``Post-
merge_selective_inference

Merged version of the selective test

Description

Merged version of the selective test

Usage

merge_selective_inference(X, k1, k2, g, ndraws = 2000, cl_fun, cl)

Arguments

X
  The data matrix of size on which the clustering is applied
k1
  The first cluster of interest
k2
  The second cluster of interest
g
  The variables for which the test is applied
ndraws
  The number of Monte-Carlo samples
cl_fun
  The clustering function used to build clusters
cl
  The labels of the data obtained thanks to the cl_fun function

Value

A list with the following elements

- pval: The resulting p-values of the test.
- adjacent: List of the adjacent clusters between k1 and k2
- pval_adj: The corresponding adjacent p-values that are merged

Examples

X <- matrix(rnorm(200),ncol = 2)
hcl_fun <- function(x){
  return(as.factor(cutree(hclust(dist(x), method = "ward.D2"), k=4)))}
cl <- hcl_fun(X)
plot(X, col=cl)
#Note that in practice the value of ndraws (the number of Monte-Carlo simulations must be higher)
test_var1 <- test_selective_inference(X, k1=1, k2=4, g=1, ndraws =100, cl_fun = hcl_fun, cl = cl)
Multimodality test for post clustering variable involvement

**Usage**

```
test_multimod(X, g, cl, k1, k2)
```

**Arguments**

- **X**: The data matrix of size on which the clustering is applied
- **g**: The variable on which the test is applied
- **cl**: The labels of the data obtained thanks to a clustering algorithm
- **k1**: The first cluster of interest
- **k2**: The second cluster of interest

**Value**

A list containing:

- **data_for_test**: The data used for the test
- **stat_g**: The dip statistic
- **pval**: The resulting p-values of the test computed with the `diptest` function

**Examples**

```r
X <- matrix(rnorm(200), ncol = 2)
hcl_fun <- function(x){
  return(as.factor(cutree(hclust(dist(x), method = "ward.D2"), k=2)))
}  
c1 <- hcl_fun(X)  
plot(X, col=c1)  
test_var1 <- test_multimod(X, g=1, k1=1, k2=2, cl = c1)  
test_var2 <- test_multimod(X, g=2, k1=1, k2=2, cl = c1)
```
**test_selective_inference**

*Selective inference for post-clustering variable involvement*

**Description**

Selective inference for post-clustering variable involvement

**Usage**

```r
test_selective_inference(
  X,
  k1,
  k2,
  g,
  ndraws = 2000,
  cl_fun,
  cl = NULL,
  sig = NULL
)
```

**Arguments**

- `X` The data matrix of size on which the clustering is applied
- `k1` The first cluster of interest
- `k2` The second cluster of interest
- `g` The variables for which the test is applied
- `ndraws` The number of Monte-Carlo samples
- `cl_fun` The clustering function used to build clusters
- `cl` The labels of the data obtained thanks to the `cl_fun` function
- `sig` The estimated standard deviation. Default is NULL and the standard deviation is estimated using only observations in the two clusters of interest

**Value**

A list with the following elements

- `stat_g`: the test statistic used for the test.
- `pval`: The resulting p-values of the test.
- `stder`: The standard deviation of the p-values computed thanks to the Monte-Carlo samples.
- `clusters`: The labels of the data.

**Note**

This function is adapted from the clusterpval::test_clusters_approx() of Gao et al. (2022) (available on Github: https://github.com/lucylgao/clusterpval)
References

Examples
X <- matrix(rnorm(200), ncol = 2)
hcl_fun <- function(x){
  return(as.factor(cutree(hclust(dist(x), method = "ward.D2"), k=2)))
} cl <- hcl_fun(X)
plot(X, col=cl)
#Note that in practice the value of ndraws (the number of Monte-Carlo simulations must be higher)
test_var1 <- test_selective_inference(X, k1=1, k2=2, g=1, ndraws = 100, cl_fun = hcl_fun, cl = cl)
Index

merge_selective_inference, 2

test_multimod, 3
test_selective_inference, 4