

Package ‘spatialRegroup’

May 11, 2026

Title Iterative Spatial Regrouping of Administrative Units by
Attributive Affinity

Version 0.1.0

Description Evaluates the statistical coherence of existing administrative partitions (e.g. inter-municipal groupings, districts) by identifying spatial units whose attributive profile is more similar to a neighbouring group than to their own. Border units are iteratively reassigned to the group they are most affine with, based on Euclidean or Mahalanobis distance computed on user-supplied numeric variables, with optional per-variable weighting and standardisation. Spatial contiguity is enforced throughout: isolated candidates are reintegrated into their original group, disconnected fragments are resolved, and empty groups are restored. Convergence is monitored via an eta-squared cohesion criterion. The resulting partition can be compared to the original administrative delineation using multilevel models, providing a quantitative measure of boundary inefficiency.

License GPL-3

Depends R (>= 4.1.0)

Encoding UTF-8

RoxygenNote 7.3.1

Imports dplyr, sf, spdep, igraph, rlang

NeedsCompilation no

Author Nicolas Ausello [aut, cre] (ORCID:
<<https://orcid.org/0000-0003-3843-0463>>)

Maintainer Nicolas Ausello <nicolasausello@yahoo.fr>

Repository CRAN

Date/Publication 2026-05-11 19:10:09 UTC

Contents

assign_best_group	2
build_neighbors	3

compute_affinity	4
compute_cohesion	5
compute_profiles	6
remove_isolates	7
spatialRegroup	8

Index	10
--------------	-----------

assign_best_group	<i>Rattacher chaque commune candidate au groupe voisin le plus proche</i>
-------------------	---

Description

Rattacher chaque commune candidate au groupe voisin le plus proche

Usage

```
assign_best_group(
  data,
  group_var,
  vars_attr,
  profiles,
  nb = NULL,
  local_profile = TRUE,
  weights = NULL,
  standardize = TRUE
)
```

Arguments

data	Un objet sf avec colonne "candidate" et "id_row"
group_var	Variable de groupe
vars_attr	Variables attributaires
profiles	Profils moyens par groupe (utilise en fallback si local_profile = FALSE)
nb	Matrice de voisinage (spdep) — obligatoire si local_profile = TRUE
local_profile	Si TRUE (default), compare au profil des membres adjacents du groupe cible plutot qu'au centroide global du groupe. Plus representatif pour les grands groupes.
weights	Vecteur de poids pour les variables (meme longueur que vars_attr)
standardize	Standardiser les variables avant calcul (default TRUE)

Value

data.frame avec id_row et best_group

Examples

```
library(sf)
make_sq <- function(x, y) {
  st_polygon(list(matrix(
    c(x, y, x+1, y, x+1, y+1, x, y+1, x, y),
    ncol = 2, byrow = TRUE
  )))
}
toy <- st_sf(
  EPCI = c('A', 'A', 'B', 'A', 'B', 'B'),
  var1 = c(1.0, 1.2, 3.8, 1.1, 3.9, 3.7),
  var2 = c(2.0, 1.9, 6.0, 2.1, 5.8, 6.1),
  geometry = st_sfc(
    make_sq(0,0), make_sq(1,0), make_sq(2,0),
    make_sq(0,1), make_sq(1,1), make_sq(2,1)
  )
)
toy[['id_row']] <- seq_len(nrow(toy))
toy[['candidate']] <- c(FALSE, FALSE, TRUE, FALSE, FALSE, FALSE)
nb <- build_neighbors(toy)
profiles <- compute_profiles(toy, 'EPCI', c('var1', 'var2'))
assign_best_group(toy, 'EPCI', c('var1', 'var2'), profiles, nb = nb)
```

build_neighbors

Construire la matrice de voisinage spatial

Description

Construire la matrice de voisinage spatial

Usage

```
build_neighbors(data, type = "queen")
```

Arguments

data	Un objet sf
type	"queen" ou "rook"

Value

Un objet nb (spdep)

Examples

```

library(sf)
make_sq <- function(x, y) {
  st_polygon(list(matrix(
    c(x, y, x+1, y, x+1, y+1, x, y+1, x, y),
    ncol = 2, byrow = TRUE
  )))
}
toy <- st_sf(
  EPCI = c('A', 'A', 'B', 'A', 'B', 'B'),
  var1 = c(1.0, 1.2, 3.8, 1.1, 3.9, 3.7),
  var2 = c(2.0, 1.9, 6.0, 2.1, 5.8, 6.1),
  geometry = st_sfc(
    make_sq(0,0), make_sq(1,0), make_sq(2,0),
    make_sq(0,1), make_sq(1,1), make_sq(2,1)
  )
)
nb <- build_neighbors(toy)
length(nb) # 6 elements (un par commune)

```

compute_affinity

Calculer les affinités attributaires des communes frontalières

Description

Calculer les affinités attributaires des communes frontalières

Usage

```

compute_affinity(
  data,
  nb,
  group_var,
  vars_attr,
  method = "euclidean",
  threshold = 0,
  weights = NULL,
  standardize = TRUE
)

```

Arguments

data	Un objet sf
nb	Matrice de voisinage (spdep)
group_var	Nom de la variable de groupe (ex: "EPCI")
vars_attr	Vecteur de noms de variables attributaires
method	Méthode de distance : "euclidean" (default) ou "mahalanobis"

threshold	Seuil d'affinite au-dela duquel une unite est candidate (default 0)
weights	Vecteur de poids pour les variables (meme longueur que vars_attr, default NULL = poids egaux). Applique apres standardisation.
standardize	Standardiser les variables avant calcul (default TRUE)

Value

Un data.frame avec dist_intra, dist_extra, affinite et candidat

Examples

```
library(sf)
make_sq <- fonction(x, y) {
  st_polygon(list(matrix(
    c(x, y, x+1, y, x+1, y+1, x, y+1, x, y),
    ncol = 2, byrow = TRUE
  )))
}
toy <- st_sf(
  EPCI = c('A', 'A', 'B', 'A', 'B', 'B'),
  var1 = c(1.0, 1.2, 3.8, 1.1, 3.9, 3.7),
  var2 = c(2.0, 1.9, 6.0, 2.1, 5.8, 6.1),
  geometry = st_sfc(
    make_sq(0,0), make_sq(1,0), make_sq(2,0),
    make_sq(0,1), make_sq(1,1), make_sq(2,1)
  )
)
nb <- build_neighbors(toy)
aff <- compute_affinity(toy, nb, group_var = 'EPCI',
  vars_attr = c('var1', 'var2'))
aff[, c('id_row', 'affinite', 'candidat')]
```

compute_cohesion

Calculer la coherence interne d'un groupement (eta-carre moyen)

Description

Mesure la part de variance inter-groupe sur la variance totale pour chaque variable attributaire, puis en fait la moyenne. Valeur entre 0 et 1 : plus elle est elevee, plus le groupement est coherent avec les variables. Equivalent a un η^2 (rapport de correlation) sans necessiter de modele mixte.

Usage

```
compute_cohesion(data, group_var, vars_attr)
```

Arguments

data	Un objet sf ou data.frame
group_var	Nom de la variable de groupe
vars_attr	Vecteur de noms de variables attributaires

Value

Un scalaire entre 0 et 1 (moyenne des η^2 par variable)

Examples

```
library(sf)
make_sq <- function(x, y) {
  st_polygon(list(matrix(c(x,y,x+1,y,x+1,y+1,x,y+1,x,y), ncol=2, byrow=TRUE)))
}
toy <- st_sf(EPCI=c('A', 'A', 'B', 'A', 'B', 'B'),
  var1=c(1.0,1.2,3.8,1.1,3.9,3.7), var2=c(2.0,1.9,6.0,2.1,5.8,6.1),
  geometry=st_sfc(make_sq(0,0),make_sq(1,0),make_sq(2,0),
    make_sq(0,1),make_sq(1,1),make_sq(2,1)))
compute_cohesion(toy, 'EPCI', c('var1', 'var2'))
```

compute_profiles	<i>Calculer les profils moyens par groupe</i>
------------------	---

Description

Calculer les profils moyens par groupe

Usage

```
compute_profiles(data, group_var, vars_attr)
```

Arguments

data	Un objet sf
group_var	Variable de groupe
vars_attr	Variabes attributaires

Value

data.frame des profils moyens par groupe

Examples

```
library(sf)
make_sq <- function(x, y) {
  st_polygon(list(matrix(
    c(x, y, x+1, y, x+1, y+1, x, y+1, x, y),
    ncol = 2, byrow = TRUE
  )))
}
toy <- st_sf(
  EPCI = c('A', 'A', 'B', 'A', 'B', 'B'),
  var1 = c(1.0, 1.2, 3.8, 1.1, 3.9, 3.7),
```

```

var2 = c(2.0, 1.9, 6.0, 2.1, 5.8, 6.1),
geometry = st_sfc(
  make_sq(0,0), make_sq(1,0), make_sq(2,0),
  make_sq(0,1), make_sq(1,1), make_sq(2,1)
)
)
toy[['candidate']] <- FALSE
profiles <- compute_profiles(toy, 'EPCI', c('var1', 'var2'))
profiles

```

remove_isolates	<i>Reintegrer les communes candidates isolees</i>
-----------------	---

Description

Reintegrer les communes candidates isolees

Usage

```
remove_isolates(data, nb)
```

Arguments

data	Un objet sf avec colonne "candidate"
nb	Matrice de voisinage

Value

data avec candidats isoles remis a FALSE

Examples

```

library(sf)
make_sq <- function(x, y) {
  st_polygon(list(matrix(c(x,y,x+1,y,x+1,y+1,x,y+1,x,y), ncol=2, byrow=TRUE)))
}
toy <- st_sf(EPCI=c('A', 'A', 'B', 'A', 'B', 'B'),
  var1=c(1.0,1.2,3.8,1.1,3.9,3.7), var2=c(2.0,1.9,6.0,2.1,5.8,6.1),
  geometry=st_sfc(make_sq(0,0),make_sq(1,0),make_sq(2,0),
    make_sq(0,1),make_sq(1,1),make_sq(2,1)))
toy[['candidate']] <- c(FALSE, TRUE, FALSE, FALSE, FALSE, FALSE)
nb <- build_neighbors(toy)
res <- remove_isolates(toy, nb)
res[['candidate']]

```

spatialRegroup

*Regroupement spatial iteratif par affinite attributaire***Description**

Regroupement spatial iteratif par affinite attributaire

Usage

```

spatialRegroup(
  data,
  group_var,
  vars_attr,
  method = "euclidean",
  threshold = 0,
  iterations = 1,
  min_group = 1,
  nb_type = "queen",
  remove_isolates = TRUE,
  weights = NULL,
  standardize = TRUE,
  local_profile = TRUE,
  cohesion_tol = 0.001,
  verbose = TRUE
)

```

Arguments

data	sf object
group_var	Variable de groupe initial (ex: "EPCI")
vars_attr	Variables attributaires pour le calcul d'affinite
method	Methode de distance : "euclidean" (defaut) ou "mahalanobis"
threshold	Seuil d'affinite pour identifier un candidat (defaut 0)
iterations	Nombre maximum d'iterations (defaut 1)
min_group	Taille minimale d'un groupe valide pour le rattachement (defaut 1)
nb_type	Type de voisinage spatial : "queen" (defaut) ou "rook"
remove_isolates	Reintegrer les candidats isoles (defaut TRUE)
weights	Vecteur de poids pour les variables attributaires (meme longueur que vars_attr, defaut NULL = poids egaux). Applique apres standardisation. Exemple : c(0.5, 0.3, 0.2) pour donner plus de poids au prix.
standardize	Standardiser les variables avant calcul de distance (defaut TRUE). Recommande pour que les variables soient comparables entre elles.

local_profile	Si TRUE (defaut), compare chaque candidat au profil des membres adjacents du groupe cible plutot qu'au centroide global du groupe. Plus representatif pour les grands groupes heterogenes.
cohesion_tol	Seuil minimal de progression de la coherence (eta ²) entre deux iterations. L'algorithme s'arrete si la progression est inferieure a ce seuil (defaut 0.001). Mettre a NULL pour desactiver ce critere d'arret.
verbose	Afficher les messages de progression (defaut TRUE)

Value

Un sf object identique a l'entree avec les colonnes supplementaires : {group_var}_regroup (affectation finale) et candidate (TRUE si l' unite a ete reclassee).

Examples

```
library(sf)

# Helper : cree un carre unitaire de coin inferieur gauche (x, y)
make_sq <- function(x, y) {
  st_polygon(list(matrix(
    c(x, y, x+1, y, x+1, y+1, x, y+1, x, y),
    ncol = 2, byrow = TRUE
  )))
}

# Grille 3x2 : 6 communes, 2 groupes (A et B)
toy <- st_sf(
  EPCI = c('A', 'A', 'B', 'A', 'B', 'B'),
  var1 = c(1.0, 1.2, 3.8, 1.1, 3.9, 3.7),
  var2 = c(2.0, 1.9, 6.0, 2.1, 5.8, 6.1),
  geometry = st_sfc(
    make_sq(0,0), make_sq(1,0), make_sq(2,0),
    make_sq(0,1), make_sq(1,1), make_sq(2,1)
  )
)

res <- spatialRegroup(
  data = toy,
  group_var = 'EPCI',
  vars_attr = c('var1', 'var2'),
  verbose = FALSE
)
res[, c('EPCI', 'EPCI_regroup', 'candidate')]
```

Index

`assign_best_group`, 2

`build_neighbors`, 3

`compute_affinity`, 4

`compute_cohesion`, 5

`compute_profiles`, 6

`remove_isolates`, 7

`spatialRegroup`, 8