Package 'LabourMarketAreas'

September 7, 2020

Type Package

Title Identification, Tuning, Visualisation and Analysis of Labour Market Areas

Version 3.2.5

Date 2020-09-05

Author Daniela Ichim, Luisa Franconi, Michele D'Alo', Guido van den Heuvel

Maintainer Luisa Franconi <franconi@istat.it>

Description Produces Labour Market Areas from commuting flows available at elementary territorial units. It provides tools for automatic tuning based on spatial contiguity. It also allows for statistical analyses and visualisation of the new functional geography.

Depends R (>= 3.5), sp

Imports maptools, data.table, rgdal, rgeos, spdep, methods

License GPL (>= 2)

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2020-09-07 17:20:03 UTC

R topics documented:

`LabourMarketAreas-package`
LabourMarketAreas

Description

Makes Travel-To-Work-Areas from commuting flow data by means of the version of the TTWA algorithm by Coombes and Bond (2008) according to the implementation carried out at Istat (Franconi, D'Alo' and Ichim, 2016) within a Europan framework of Labour Market Area development (Franconi, Ichim and D'Alo' 2017). The whole process of Labour Market Areas development has been taken care of (regionalisation algorithm, creation of shape file, assignment of names, fine tuning of the output of the algorithm, quality assessment and sensitivity analysis.).

Details

Package:	LabourMarketAreas
Type:	Package
Version:	3.2.5
Date:	2020-09-05
License:	GPL (>=2)

Labour market areas (LMAs) are sub-regional geographical areas where the bulk of the labour force lives and works, and where establishments can find the main part of the labour force necessary to occupy the offered jobs. They are functional regions that stem from the aggregation of elementary geographical units (municipalities, census output areas, etc.) on the basis of their level of spatial interaction measured by commuting to work flows through quantitative methods. The guiding idea is to maximise the flow inside the area (internal cohesion) and minimise it outside (external separation) according to a predefined rule.

The package is based on original script by Guido van den Heuvel at Statistics Netherlands and further developed at the Italian National Statistical Institute (Istat) to implement the algorithm described in Coombes and Bond (2008) - a variation of the seminal paper by Coombes et al. (1986). See Franconi, D'Alo'and Ichim (2016) for the full description of this implementation and Franconi, Ichim and D'Alo' (2017) for an overview of project.

The algorithm is a rule based algorithm that stops when all areas satisfy the rule.

Every area is characterised by the number of commuters living in it, by the number of commuters that go there to work (called number of jobs/workers) and by those commuters that live and work in the same area. The ratios between these quantities define the concept of self-containment. We are interested in the minimum self-containment i.e. the minimum between the Supply side self-containment and the Demand side self-containment (see Coombes and Bond, 2008). According to Coombes and Bond (2008) an area is a Labour Market Area (LMA) if it satisfies the validity condition (see Coombes and Bond, 2008). Such validity depends on the number of commuters living in the area, the minimum self-containment and four parameters chosen by the user (see function findcluster).

The package produces LMAs for the country/region for which commuting flows are available at basic territorial level (municipality, province, census output areas, etc.). We call this basic territorial level community. If the names of the communities are provided, the package allows to assign names to each LMA (see function AssignLmaName). The package allows also to plot a cartographic map of the produced LMAs given the shape files of the communities (see function PlotLmaCommunity). Finally the package identifies enclaves (communities that are not contiguous with the rest of the LMA they belong to) and perform a re-assignment of such community based on contiguity and cohesion (see the function FineTuning).

The main output of the function findcluster is the set of LMAs produced by the algorithm, the so called a partition. There are three dimensions that characterise the partition: the dimension of the starting configuration i.e. number of communities in the country/region of interest, the dimension of the final configuration, i.e. the number of LMAs identified by the algorithm using the chosen parameters, and the flows between LMAs. These three dimensions need to be represented in the solution of the algorithm; this is done through a list of three data.table the lma (or clusterData) data structure. Such lma data structure is the cornerstone of the whole package as it is the input or the output of many functions inside the package.

The work was partially funded by Eurostat Grant "EU-TTWA method: improvements, documentation and sharing knowledge activities" awarded to Istat the Italian National Statistical Institute (http://www.istat.it/en/archive/182743). This is part of a system of grants in the EU with the aim of supporting the ongoing development of a methodology for the creation of LMA and test the application of methods nationally in the participating National Statistical Institutes. The goal is to arrive at a harmonised EU-wide definition for labour market areas. More info at:

https://ec.europa.eu/eurostat/cros/content/labour-market-areas_en

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo' and Guido van den Heuvel

Maintainer: Luisa Franconi <franconi at istat.it>

References

[1] Coombes, M. e Bond, S. (2008). Travel-to-Work Areas: the 2007 review. London: Office for National Statistics, 2008.

https://ons.maps.arcgis.com/home/item.html?id=5efb4f1c1a114fef8f74f6f0347c10e8

[2] Coombes, M., Casado-Diaz, J.M., Martinez-Bernabeu, L. e Carausu, F. (2012). Study on comparable labour market areas: final research report. 17 October 2012. Eurostat- Framework contract no:6001. 2008.001 - 2009.065, Specific contract no:50405.2010.004 - 2011.325.

[3] Coombes, M.G., Green, A.E. e Openshow, S. (1986). An efficient algorithm to generate official statistics report areas: the case of the 1984 Travel-to-Work Areas in Britain. The Journal of Operational Research Society, Vol. 37, No. 10, pp. 943-953.

[4] Franconi, L., D'Alo', M. and Ichim, D. (2016). Istat Implementation of the algorithm to develop Labour Market Areas. Available at

http://www.istat.it/en/files/2016/03/Description-of-the-LabourMarketAreas-algorithm.pdf.

[5] Franconi, L., Ichim, D. and D'Alo', M. (2017). Labour Market Areas for territorial policies: tools for a European approach. Statistical Journal of the IAOS, Vol. 33, No. 3, pp. 585-591.

https://content.iospress.com/articles/statistical-journal-of-the-iaos/sji160343

[6] Istat (2015). La nuova Geografia dei Sistemi Locali. (in italian). Available at

http://www.istat.it/it/files/2015/10/La-nuova-geografia-dei-sistemi-locali.pdf

---

`AddStatistics` *AddStatistics*

Description

Function to evaluate statistics at LMA level based on data at community level.

Usage

```
AddStatistics(statData, comID.file, lma, comID.lma)
```

Arguments

`statData`	data.frame or data.table containing the ID of the communities and the numerical variables to be summed at LMA level.
`comID.file`	character: name of the variable containing the community ID in the statData object.
`lma`	list of three data.tables: clusterList, LWClus and marginals. See function findClusters.
`comID.lma`	character: name of the variable containing the community ID in the lma object.

Details

This function can be used in general to compute several statistics at LMA level provided data at community level is provided. This function sums the values at community level to obtain the corresponding value at LMA level.

Value

data.table containing the LMA ID and the summed numerical variables from statData.

Author(s)

Ichim, D., Franconi, L., D'Alo, M.

Examples

```
# compute population totals at LMA level from population values at community level.
## Not run:
out<- findClusters(LWCom=Brindisi, minSZ=1000,minSC=0.6667,tarSZ=10000,tarSC=0.75,
verbose=TRUE)
AddStatistics(shpBrindisi@data[,c("PRO_COM","POP2001")], "PRO_COM",out$lma,"community" )

## End(Not run)
```

`AssignLmaName` *AssignLmaName*

Description

Given the names of the original communities (elementary area or municipality) as input, this function assigns the name to each labour market area given its code. The assigned LMA name corresponds to the name of the community having the highest number of jobs (incoming commuters) among all the communities in the corresponding LMA. If more than one community shares the maximum number of jobs, the first one is taken. In order to differ the community name by the LMA with the same name, the community is expressed as first letter in uppercase and the remaining letters in lowercase, whereas the LMA name is all in uppercase.

Usage

```
AssignLmaName(LWCom,lma,ComNames)
```

Arguments

`LWCom`	data frame/data.table of commuting data (see for example Sardinia).
`lma`	list of three data tables. It is the output of the function findClusters. It contains the three data.tables clusterList, LWClus, marginals.
`ComNames`	data frame/data.table containing two variables: code, integer representing the id of the community and com.name, character containing the community name. The code must be positive.

Value

`lma` list of three data.tables. It contains the three data.table clusterList, LWClus, marginals with the new added columns related to the names of the communities and their corresponding labour market areas. The added columns are character type. In clusterList the new variables are com.name, lma.name, in LWCom the variables are lma.name.live, lma.name.work and in marginals the new added variable is lma.name.

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

See Also

findClusters

`AssignSingleComToSingleLma`
AssignSingleComToSingleLma

Description

This function assigns a community to a given labour market area. It simulates a manual assignment. It might be used inside the fine tuning process.

Usage

```
AssignSingleComToSingleLma(lma,comID,lmaID,dat)
```

Arguments

`lma`	The partition to be modified. A list of three components named: clusterList, LWClus and marginals, respectively. See the output of the findClusters function.
`comID`	The identifier of the community to be assigned.
`lmaID`	The identifier of the labour market area where comID has to be assigned.
`dat`	The initial commuting matrix between communities. A data.table containing the variables community_live, community_work and amount.

Value

The new labour market partition. A list of three components named:

`clusterList`	A data.table with three variables: community, cluster, residents
`LWClus`	A data.table with three variables: cluster_live, cluster_work, amount
`marginals`	A data.table with three variables: cluster, amount_live, amount_work

Note

This function can be called before the call to the function AssignLmaName.

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

See Also

findClusters

`BindPiecesLma` *BindPiecesLma*

Description

This function (row) binds two local labour market areas structures. It deletes the communities that are registered twice (or more) in both structures. It deletes the communities that are not registered in the input flows data frame/data.frame. LMA Ids that are registered in both structures are deleted as well.

In case there are common LMA ids, those of input2 are summed to the maximum LMA id of input1.

Usage

```
BindPiecesLma(input1,input2,LWCom)
```

Arguments

`input1`	list of three data.table: clusterList, LWClus and marginals. See function findClusters.
`input2`	list of three data.table: clusterList, LWClus and marginals. See function findClusters.
`LWCom`	data frame/data.table containing the commuting flows information. Three variables: community_live, community_work and amount. See function findClusters.

Value

List of five components.

`com.twice.1`	numeric vector; communities ids that are registered twice (or more) in the first input data structure
`com.twice.2`	numeric vector; communities ids that are registered twice (or more) in the second input data structure
`coms.not.in.flows`	numeric vector; communities ids that are registered in either input data structures, but not in the commuting flows data.table

`LMAtwice` numeric vector; LMA ids that are registered in both input data structures. These LMAs are not included in the new structure.

`lma` a list of three data.tables: clusterList, LWClus and marginals. Information on commuters, residents and workers is computed using the input commuting flows data frame/data.table.

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

See Also

findClusters

`Brindisi` *Brindisi*

Description

data frame/data.table containing the commuting flows between municipalities in Brindisi province from Italian Population Census (the complete data set http://www.istat.it/it/archivio/157423). Each row corresponds to an observation i.e. a flow and each column corresponds to a variable. The variables are: community_live, community work and amount, in this order. The meaning of the variables is the following: community_live, integer, id number of the community (elementary territorial unit) where the commuter/commuters live, community_work, integer, containing the id number of the community (elementary territorial unit) where the commuter/commuters work and amount, numeric containing the number of commuter/commuters commuting between community_live and community_work (direction is important).

Usage

```
data(Brindisi)
```

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

`CompareLMAsStat` *CompareLMAsStat*

Description

This function computes several statistics to be used for comparing partitions, e.g. for parameter selection.

Usage

```
CompareLMAsStat(list.lma,dat)
```

Arguments

`list.lma`	list of LMAs to be compared. Each component of the list is a list of at least two components: lma and param (see the output of the function findCluster). The lma component is a list of three data.tables: clusterList, LWClus and marginals (the names should have not been assigned; otherwise use function DeleteLmaName). The param is a numeric vector containing the parameters of the corresponding LMAs, i.e. minSZ, minSC, tarSZ, tarSC.
`dat`	data frame/data.table with the original commuting flows between communities. It contains three columns: community_live, community_work and amount.

Value

A matrix containing the quality statistics computed by the StatClusterData function.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo'

See Also

findClusters, StatClusterData

`copyClusterData` *copyClusterData*

Description

This function copies (in data.table parlance) the three components of lma, i.e. clusterList, LWClus and marginals. See function findClusters.

Usage

```
copyClusterData(lma)
```

Arguments

lma	list of three data.tables: clusterList, LWClus and marginals. See function findClusters.

Value

list of three. It contains the three data tables clusterList, LWClus, marginals. They are copies of the input data.tables.

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

See Also

findClusters

CreateClusterData *CreateClusterData*

Description

This function creates a labour market areas structure, i.e. a list of three data.tables: clusterList, LWClus and marginals given a set of commuting data.

Usage

```
CreateClusterData(LWCom,residents=NULL)
```

Arguments

LWCom	input data frame/data.table of commuting data (see for example Sardinia).
residents	A data.table with two columns: Code and residents. Code is the community Id while residents represents the amount of occupied persons living in the community. If NULL, the residents data.table is computed using the commuting flows dataset. The missing values are not allowed: communities with no residents present zero values.

Details

If needed, the fictitious community (see Franconi, D'Alo' and Ichim, 2016 for definition and details) should be a-priori included in both residents and commuting flows datasets.

Value

A list of three components:

clusterList:	data.table with three variables: community, cluster and residents. Each community is assigned to a cluster. The clusters ID is a numerical sequential vector, generated independently on the community IDs values.
LWClus:	a data.table representing the commuting flows between clusters. The three variables are cluster_live, cluster_work and amount
marginals:	a data.table representing the cluster characteristics. The three variables are cluster, amount_live (residents) and amount_work (workers).

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

References

[1] Franconi, L., D'Alo', M. and Ichim, D. (2016). Istat Implementation of the algorithm to develop Labour Market Areas. Available at

http://www.istat.it/en/files/2016/03/Description-of-the-LabourMarketAreas-algorithm.pdf.

See Also

findClusters

CreateLMAShape *CreateLMAShape*

Description

Starting from the communities shape files, the function creates the shape files of labour market areas.

Usage

```
CreateLMAShape(lma,comIDs,lmaIDs,shp_com,dsn,shp_com_name,id_shp_com,outd,outf,bf,po)
```

Arguments

lma	List (three data.tables) containing all defining information of the LMAs. See the output of the function findClusters. The names of the clusters have been assigned, see AssignLmaName function.
comIDs	Name of the variable containing the community identifier in the lma input.
lmaIDs	Name of the variable containing the labour market areas identifier in the lma input.

shp_com	Spatial polygon (R object) containing the shape file of the communities. Defaults to NULL. In case both shp_com and shp_com_name are provided, only the first one is considered.
dsn	character; data source name specifying the directory (path) in which the shp_com_name file is stored.
shp_com_name	character; the file name of the communities shape file. Defaults to NULL. In case both shp_com and shp_com_name are provided, only the first one is considered.
id_shp_com	character; name of the variable containing the communities identifier in the shp_com object or in the shp_com_name file.
outd	character. The path where to save the shape files. Defaults to NULL.
outf	character. The file name where to save the shape files. Defaults to NULL. If it is provided, the outdir should be provided, too.
bf	character. The bmpfile file name (including the path) of the bitmap file where to save the cartographic map of the communities and labour market areas together. Defaults to NULL.
po	Graphical parameters (plot_opt) to be used when bmpfile is not NULL. The parameters are: background colour of the map, line width of the lma borders, line type of the lma borders, lma names color, line width for the lma names, line type for the lma names, cex factor for the lma names, font for the lma names. Defaults to c("green", 1, 2, "red", 1, 2, 0.8, 2).

Value

A list of three components:

shp_lma	spatial polygon containing the shape file of lma
comID.in.LMA.not.in.SHP	vector of communities registered in lma but not in the input shape object/file.
comID.in.SHP.not.in.LMA	vector of communities registered in the input shape object/file but not in the lma.

Note

This function has to be applied on a labour market partition whose names are assigned. Note that some communities present in the shape files may not be available in the lma structure. These missing communities can be found in the zero.list and ComNotAssigned. They could also be structurally missing i.e. could not be present in the initial commuting data. Note also that if the option of providing an output file is activated then the usage of an existing file name in the selected outdir will generate an error.

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

See Also

findClusters, AssignLmaName

`DeleteLmaName` *DeleteLmaName*

Description

This function deletes the variables corresponding to names of communities and clusters.

Usage

```
DeleteLmaName(lma)
```

Arguments

`lma`	list of three data.tables: clusterList, LWClus and marginals. See function findClusters.

Value

list of three. It contains the three data.tables clusterList, LWClus, marginals. From clusterList the deleted variables are com.name, lma.name, from LWCom the deleted variables are lma.name.live, lma.name.work and from marginals the deleted variable is lma.name.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo'

See Also

findClusters

`determineCohesion` *determineCohesion*

Description

Computes the cohesion matrix.

Usage

```
determineCohesion(LWClus, marginals)
```

Arguments

`LWClus`	data table containing the flows between the clusters.
`marginals`	data table containing the main characteristics of the current set of clusters.

Value

The cohesion matrix between clusters.

Note

This function is called and used internally by the main function findClusters through the function determineRegroupList.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo' and Guido van den Heuvel

References

[1] Coombes, M.G., Green, A.E. and Openshow, S. (1986). An efficient algorithm to generate official statistics report areas: the case of the 1984 Travel-to-Work Areas in Britain. The Journal of Operational Research Society, Vol. 37, No. 10, pp. 943-953.

[2] Franconi, L., D'Alo' M. and Ichim, D. (2016). Istat implementation of the algorithm to develop Labour Market Areas.

See Also

determineRegroupList

`determineRegroupList` *determineRegroupList*

Description

This function determines, for each cluster in LWClus, the cluster with which that cluster is most closely associated. First the cohesion between all pairs of clusters is calculated; then the cohesion is sorted for each cluster and only the last (i.e., highest) one is selected.

Usage

```
determineRegroupList(LWClus, marginals)
```

Arguments

`LWClus`	data table containing the flows between the clusters.
`marginals`	data table containing the main characteristics of the cluster.

Details

This function determines, for each pair of cluster in LWClus, the cohesion; then it is sorted and only the last (i.e., highest) one is selected. The highest value it associated with the dominant cluster for the dissolved cluster.

Value

Integer: the candidate cluster identifier.

Note

This function is called and used internally by the main function findClusters.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo' and Guido van den Heuvel

References

[1] Franconi, L., D'Alo' M. and Ichim, D. (2016). Istat implementation of the algorithm to develop Labour Market Areas.

See Also

findClusters

---

dissolveCluster *dissolveCluster*

Description

This function dissolves a selected cluster into its constituent communities. Such communities are given temporary cluster IDs. Then clusterList is updated with the temporary IDs and the number of commuters (LWClus) and the cluster structure (marginals) are recomputed.

Usage

```
dissolveCluster(clusterData, cluster, LWCom)
```

Arguments

`clusterData`	list of 3 data.tables defining all the elements of the current set of clusters.
`cluster`	integer: id of the selected cluster to be dissolved into its constituent communities.
`LWCom`	data frame/data.table containing the commuting data.

Value

`clusterData`	list of 3 data tables defining all the elements of the current set of clusters. The ID includes negative values corresponding to the dissolved cluster.

Note

This function is called and used internally by the main function findClusters.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo and Guido van den Heuvel

References

[1] Franconi, L., D'Alo' M. and Ichim, D. (2016). Istat implementation of the algorithm to develop Labour Market Areas.

See Also

findClusters

`dissolveClusterSel` *dissolveClusterSel*

Description

This function dissolves a selected cluster into its constituent communities. Such communities are given temporary cluster IDs. Then clusterList is updated with the temporary IDs and the number of commuters (LWClus) and the cluster structure (marginals) are recomputed.

Usage

```
dissolveClusterSel(clusterData, cluster, lwcom)
```

Arguments

`clusterData`	list of 3 data.tables defining all the elements of the current set of clusters.
`cluster`	integer: id of the selected cluster to be dissolved into its constituent communities.
`lwcom`	data frame/data.table containing the selected commuting data.

Value

`clusterData`	list of 3 data tables defining all the elements of the current set of clusters. The ID includes negative values corresponding to the dissolved cluster.

Note

This function is called and used internally by the main function findClusters.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo and Guido van den Heuvel

References

[1] Franconi, L., D'Alo' M. and Ichim, D. (2016). Istat implementation of the algorithm to develop Labour Market Areas.

See Also

findClusters

`EqualLmaPartition` *EqualLmaPartition*

Description

This function tests whether two partitions are equal.

Usage

```
EqualLmaPartition(lma1, lma2)
```

Arguments

`lma1`	List (three components) containing all defining information of the partition. The clusterList component should contain variables community and LMA. See findClusters function.
`lma2`	List (three components) containing all defining information of the partition to be compared with the previous one.The clusterList component should contain variables community and LMA.See findClusters function.

Value

A logical value.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo'

See Also

findClusters

findClusters *findClusters*

Description

This function builds labour market areas (LMAs) starting from commuting data between communities i.e. elementary territorial units (municipalities, census output areas, provinces, etc.). The function implements the algorithm described in Coombes and Bond (2008) according to the implementation detailed in Franconi, D'Alo' and Ichim (2016).

Usage

```
findClusters(LWCom, minSZ, minSC, tarSZ, tarSC,
verbose = F, sink.output = NULL,trace=NULL,
PartialClusterData=NULL, idcom_type=NULL)
```

Arguments

LWCom	data frame/data.table containing the commuting data. Each row corresponds to an observation i.e. a flow and each column corresponds to a variable. The variables are named: "community_live", integer contains the id number of the elementary zone of residence, "community_work", integer, contains the id number of the working community (elementary zone of work), "amount", integer/numeric contains the number of commuters commuting between the "community_live" and the "community_work" (direction is important). Missing values (NAs) are not allowed. The community id must be positive. Only positive flows are present in the data frame. See Sardinia.
minSZ	integer specifying the parameter containing the acceptable minimum size of an area in terms of occupied persons. Must be positive.
minSC	numeric in the interval (0,1) specifying the parameter representing the acceptable minimum self-containment of an area. Usually values range from 0.6 to 0.7.
tarSZ	integer specifying the parameter containing the target size of an area in terms of occupied persons. It must be greater than minSZ.
tarSC	numeric in the interval (0,1) specifying the parameter representing the target self containment of an area. It must be greater than minSZ. Usually values range 0.75 to 0.9.
verbose	logical. If TRUE the iteration number and the minimum validity are printed on the screen together with warning messages. Default is FALSE.
sink.output	character string containing the name of the .txt file that will contain optional information for each iteration of the algorithm. Default is NULL i.e. the sink function is not activated.
trace	integer. If not NULL (default value) and if the number of (internal) iterations is multiple of it, an intermediate output is saved in the file intermclusterData.Rdata in the current working directory. The intermediate output is a list of two elements: a clusterData object and the vector of parameters.

`PartialClusterData`
: labour market areas structure (clusterData) representing the starting point of the iterative algorithm.

  The labour market areas structure is a list of three components: clusterList (whose variables are: community, cluster and residents), LWClus (variables: cluster_live, cluster_work and amount) and marginals (variables: cluster, amount_live and amount_work). Defaults to NULL.

  If it is not NULL, the information related to the process used to derive this structure is not registered in the output of the function).

`idcom_type`
: character. If not NULL (default value) the identification code of the community is a character and not an integer.

Value

The output of the function is a list of lists with components:

`lma`
: List of three data.tables: it contains all the information on the partitioning of the initial communities stemming from the input data frame LWCom into a set of labour market areas. Each data frame contains a dimension of the partition: the initial list of communities (municipalities or elementary areas), the relationships between the labour market areas and their structural characteristics.

  The three data.tables are:

  clusterList: data.table containing the allocation of each community to the corresponing Labour Market Areas (LMA). It includes three variables: *community*, integer containing the id of the community, *cluster*, integer containing the id of the labour market areas, and *residents*, integer/numeric containing the number of commuters who are residents in the corresponding community.

  LWClus: data.table containing the flows between the LMA. It includes three variables: *cluster_live*, integer representing the id number of the labour market area of residence, *cluster_work*, integer representing the id number of the labour market area of work and *amount*, numeric representing the number of employee commuting from cluster_live to cluster_work (the direction is important).

  marginals: data.table containing the main characteristics of the labour market areas. The variables representing such characteristics are: *cluster*, integer containing the id number of the labour market area, *amount_live* numeric, number of employees who are residents in the LMA (i.e. who live in the LMA), *amount_work* numeric, number of employees working in LMA regardless of where they live (this variable is also known as workers or jobs).

`lma.before0`
: List with the same structure as lma (above) containing the result of the algorithm before the assignment of the communities belonging to the reserve list to the dominating labour market areas (if it exists). The reserve list is the LMA with id=0. The communities belonging to it can be investigated through the data.table clusterList.

`reserve.list`
: List of lists. Communities that do not improve the value of the validity when assigned to the dominating cluster or that do not have a dominating cluster are put into the reserve list. Each list contains a character string with information on:

the type reason for being assigned to the reserve list; possible values are: "A", "B", "C", "D", "E", "F". Please refer to Franconi, D'Alo' and Ichim (2016) for details on the description of such cases.

the iteration in which the community was assigned to the reserve list;

the id of the dissolved cluster i.e. the cluster to which the community belonged before being assigned to the reserve list;

the value of the validity of such cluster (in character format);

the id of the community assigned to the reserve list;

the community belonging to the cluster being dissolved with the second lowest external relationships value see Franconi, D'Alo' and Ichim (2016) (if available otherwise NULL).

`comNotAssigned` List. Component: integer containing the id of the community in the reserve list that the algorithm was not able to assign to any existing cluster. NULL otherwise.

`zero.list` List of four objects: they contain information on communities (elementary areas or municipalities) that could not be processed by the algorithm for various reasons; either the number of commuters resident in it is 0 or the number of workers/jobs is 0 or the community has no interaction with any other community. In such cases the algorithm eliminates these communities from the initial list and let the user the choice to allocate them at a later stage (see function AssignSingleComToSingleLma).

Communities: integer containing the ids of the communities that could not be processed by the algorithm.

LWCom: data.table containing the flows involving the above communities. The data.table is in a sense a subset of the initial data.table containing the commuting data. Its variables are community_live, integer containing the id of the community where the commuters live, community_work, integer containing the id of the community where the commuters work, amount, numeric, containing the number of commuters commuting from community_live to community_work.

Residents, data frame containing the variables Code, integer representing the id of the community and residents, integer representing the number of employees who are resident in the community.

Workers, data frame containing the variables Code, integer representing the id of the community and workers, integer/numeric representing the number of commuters working in the community (jobs).

`communitiesMovements`

data.table with two columns: *community* and *moves*.

community: integer. It represents the community ID;

moves: integer. It represents the number of times a community has changed cluster. The movement toward the reserve list is not computed.

`param` vector containing the parameters used to apply the algorithm, i.e. minSZ, minSC, tarSZ, tarSC.

`idcom_rel` data.table containing the list of original id community (character) and their corresponding numerical labels created and used inside the algorithm.

Note

Note that everytime that the idcom_type is not NULL in all the output of the function the community identifier will be character and not integer. In anycase in the sink file and in the PartialClusterData component the community identifier will still be integer and not character.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo' and Guido van den Heuvel

References

[1] Franconi, L., D'Alo' M. and Ichim, D. (2016). Istat implementation of the algorithm to develop Labour Market Areas.

See Also

LMAwrite

Examples

```
## Not run:
out<- findClusters(LWCom=Brindisi, minSZ=1000,minSC=0.6667,tarSZ=10000,tarSC=0.75,
verbose=TRUE)

## End(Not run)
```

`FindContig` *FindContig*

Description

Given a territorial object (either an LMA, formed by a single community, or a polygon) the function determines its contiguous labour market areas.

Usage

```
FindContig(type = "poly", lma, contig.matrix, isolated)
```

Arguments

`type`	Character indicating which type of objects should be treated. It may be "poly" (polygons) or "lma" (local labour market areas). Defaults to "poly".
`lma`	A list of three components clusterList, LWClus, marginals. See the function FindClusters.
`contig.matrix`	matrix: the (spatial) contiguity matrix to be used (see the function FindIsolated).

`isolated` Information on the isolated objects. If type is "poly", isolated is the association matrix between communities and polygons. See the poly.com.linkage component in the output of FindIsolated function. If type is "lma", isolated is the vector of labour market areas identifiers. See the lma.unique component in the output of the FindIsolated function.

Value

If type="poly", the output is a list of two components:

`list.contig.poly`
: list containing the IDs of the contiguous labour market areas of each community (polygon). The list names are the communities identifiers.

`com_no.LMA.neigh`
: character. The names of the communities having only polygons as neighbours.

If type="lma", the output is a list containing the IDs of the contiguous labour market areas of each given lma. The list names are the communities identifiers. This option should be used for the labour market areas with a unique community.

Note

This function should be used to identify the neighbours of the labour market areas having an unique community (type="lma") or to identify the labour market areas which are contiguous to a polygon associated to a unique community (the case of a community having an enclave outside its own territory).

If type="poly", there is no special ordering of the contiguous LMAs.

If type="lma", the contiguous LMAs are ordered in decreasing order of commuters who are resident in the LMA.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo'

See Also

findClusters, FindIsolated

`FindIsolated` *FindIsolated*

Description

A labour market area is defined as isolated when there are no (spatial) neighbours or there is a unique community inside it. A polygon is defined as isolated when there are no (spatial) neighbours (e.g. a small island which is part of a community in the main land is isolated). This function identifies the isolated labour market areas and their isolated polygons. The contiguity between two objects (labour market areas or polygons) is based on spatial relationships, not on their commuting flows.

Usage

```
FindIsolated(lma,lma_shp=NULL,lma_shp_path,
lma_shp_name,com_shp=NULL,com_shp_path,
com_shp_name, id_com)
```

Arguments

lma	List (three components) containing all defining information on the labour market areas. See findClusters function. The names of the lma have already been assigned (see AssignLmaName).
lma_shp	spatial polygon object corresponding to the labour market areas. See function CreateLmaShape. Defaults to NULL.
lma_shp_path	character. The path where the shape files of the labour market areas are saved. See function CreateLmaShape. Defaults to NULL.
lma_shp_name	character. The file name where the shape files of the labour market areas are saved. See function CreateLmaShape. Defaults to NULL.
com_shp	spatial polygon object corresponding to the communities. Defaults to NULL.
com_shp_path	character. The path where the shape files of the communities are saved. Defaults to NULL.
com_shp_name	character. The file name where the shape files of the communities are saved. Defaults to NULL.
id_com	character. The field name of the variable containing the community ID in the communities spatial polygon or in their shape files.

Value

A list of two components.

`isolated.lma` A list of three components: contig.matrix.lma, lma.unique and lma.nolink.

contig.matrix.lma: matrix. The contiguity matrix of the given labour market areas.

lma.unique: data.table. It has two columns: lma.unique.ID and lma.unique.name. lma.unique.ID: integer, the identifier of the unique labour market areas. lma.unique.name: character, name of the unique labour market area. The unique areas are the areas with a unique community.

lma.nolink data.table with two columns: lma.nolink.ID and lma.nolink.name. lma.nolink.ID integer, the identifier of the labour market area with no links. lma.nolink.name: character, the name of the labour market area with no links.

`isolated.poly` A list of three components: contig.matrix.poly, poly.com.linkage and poly.nolink.

contig.matrix.poly: matrix. It is the contiguity matrix of the polygons.

poly.com.linkage data.table with two columns, community and Polygon. It represents the association between communities and polygons.

poly.nolink: data.table with two colums: poly.nolink.ID and poly.nolink.name. poly.nolink.ID integer: the identifiers of the no-linked polygons. poly.nolink.name character: the name of the labour market area with no-linked polygons.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo'

See Also

findClusters, CreateLmaShape

`FineTuning` *FineTuning*

Description

This function assigns enclaves (i.e.communities - or part of them - that are not contiguous to the rest of the LMA they belong to) to labour market areas, based on contiguity principle and the cohesion.

Usage

```
FineTuning(dat, out.ini, list.contiguity)
```

Arguments

`dat`	The commuting flows between communities. See function findCluster.
`out.ini`	A list of three components: clusterList, LWClus and marginals. See function findClusters. The names should not have been assigned. Use function DeleteLmaName.
`list.contiguity`	list. Each component of the list is a vector indicating the neighbouring labour market areas. The names of the list are the communities IDs.

Details

The algorithm (function findCluster) assigns communities to clusters based on a rule. There is no check that communities in clusters are contiguous. For this reason a fine tuning of the initial result is needed (the function FineTuning). There is no ordering of the neighbours.

Value

A list of two components:

`tunned.lma`	list of three components clusterList, LWClus and marginals containing the result of the fine tuning procedure. See function findClusters
`not.tunned.commID`	a vector of communities IDs that were not assigned based on a contiguity principle; e.g. those communities having no flows with their neighbours.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo'

getLeastSelfContained *getLeastSelfContained*

Description

This function determines the validity for all clusters as well as the minimum value i.e. the value for which the "X-equation" is minimal (see [1] and [2]). This minimum corresponds to the selected cluster to be dissolved. If more than one cluster share the same minimum value the first will be chosen.

Usage

```
getLeastSelfContained(LWClus, marginals, minSZ, minSC, tarSZ, tarSC)
```

Arguments

LWClus	data table containing the flows between the clusters.
marginals	data table containing the structural characteristics of the clusters.
minSZ	numeric, parameter indicating the minimum size of the cluster in order to be an acceptable LMA.
minSC	numeric, parameter indicating the minimum self containment of the cluster in order to be an acceptable LMA.
tarSZ	numeric, parameter indicating the target size of the clusters.
tarSC	numeric, parameter indicating the target self containment of the clusters.

Value

A list of two components:

minivalout	data.table containing the identifier of cluster attaining the minimum validity and its corresponding value.
LWSelf	data.table containing the validity computations for each cluster.

Note

This function is called and used internally by the main function findClusters.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo' and Guido van den Heuvel

References

[1] Coombes, M. e Bond, S. (2008). Travel-to-Work Areas: the 2007 review. London: Office for National Statistics, 2008.

[2] Franconi, L., D'Alo' M. and Ichim, D. (2016). Istat implementation of the algorithm to develop Labour Market Areas.

See Also

findClusters

`LMAwrite` *LMAwrite*

Description

This function saves the lists composing the output of the lma package into separate data frames as .RData. The files are saved in the path_wd directory. The main output - the first and second list containing, respectively, the characteristics of the created labour market areas and the characteristics of the areas before the assignment of the reserve list - are also saved in a .csv file.

Usage

```
LMAwrite(out, path_wd = NULL, suff = NULL)
```

Arguments

`out`	list of lists containing the output of the lma package.
`path_wd`	character containing the path of the directory where the output ought to be saved. Default is NULL, working directory.
`suff`	character containing the suffix to be added to the name of the saved output files. Default is NULL.

Note

This function has to be applied on a labour market partition whose names are not assigned otherwise it will generate an error.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo' and Guido van den Heuvel

---

`mergeCluster` *mergeCluster*

Description

This function updates clusterData by merging cluster1 and cluster2.

Usage

```
mergeCluster(clusterData, cluster1, cluster2)
```

Arguments

clusterData	List (three components) containing all defining information on the current clusters found by the algorithm.
cluster1	Cluster ID to be merged. An unique positive integer.
cluster2	cluster ID to be merged. An unique positive integer.

Value

A cluster data object, i.e. a list of three components:

ClusterList	data.table of three columns: community, cluster, residents
LWClus	data.table of three columns; cluster_live, cluster_work and amount
marginals	data.table of three columns: cluster, amount_live, amount_work

Note

This function is called and used internally by the main function findClusters.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo' and Guido van den Heuvel

See Also

findClusters

---

names.Brindisi *names.Brindisi*

Description

data frame/data.table; it contains Code, integer representing the id of the community (elementary area or municipality) and com.name, character, containing the community name. The Code must be positive. In order to differ the community name by the lma with the same name, the community is expressed as first letter in uppercase and the remaining letters in lowercase, whereas the lma name is all in uppercase.

Usage

```
data(names.Brindisi)
```

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

names.Sardinia *names.Sardinia*

Description

data frame/data.table; it contains Code, integer representing the id of the community (elementary area or municipality) and com.name, character, containing the community name. The Code must be positive. In order to differ the community name by the lma with the same name, the community is expressed as first letter in uppercase and the remaining letters in lowercase, whereas the lma name is all in uppercase.

Usage

```
data(names.Sardinia)
```

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

PlotLmaCommunity *PlotLmaCommunity*

Description

This function visualizes/plots the LMAs containing given communities.

It may be used to compare two partitions or to see the assignment of a set of communities during the iterations of the algorithm (object intermClusterData when trace is not NULL in the findClusters function).

Usage

```
PlotLmaCommunity(list.lma, lmaIDs, communityID, shp_com, id_shp,
bmpfile, col.vec)
```

Arguments

list.lma	List of two labour market areas structures (lma). The two components of the list are lists with components clusterList, LWClus and marginals representing the two different partitions to be compared. The names of the communities and LMAs have already been assigned before running this function; See functions findClusters and AssignLmaName. If the list has length greater than 2, only the first two partitions will be compared.
lmaIDs	The name of the variable identifying the LMA in list.lma.
communityID	Positive integer vector, the IDs of the communities under study.

shp_com	Spatial polygon data frame containing the shape files of the communities.
id_shp	The name of the variable identifying the communities in shp_com spatial polygon data frame.
bmpfile	character: the name of the bmp file where to save the images complete with its path.
col.vec	character vector of three color names. The first one corresponds to the communities under investigation; the second color corresponds to the communities which are present in the first partition, but not in the second one; the third color corresponds to the communities which are present in the second partition, but not in the first one. Defaults to c("red","orange","yellow").

Value

logical indicating whether the communities belong to the same clusters. The identity is checked using the cluster ID.

The graphic is saved in the specified path/file.

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

See Also

findClusters, AssignLmaName

`Qmodularity` *Qmodularity*

Description

This function computes the Q-modularity index for a given partition.

Usage

```
Qmodularity(lma)
```

Arguments

lma	list of three data.frames: clusterList, LWClus and marginals. See function findClusters.

Value

numeric value, the Q-modularity index for the given partition.

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

References

[1] Y. Liu, Q. Liu, and Z. Qin, "Community Detecting and Feature Analysis in Real Directed Weighted Social Networks", Journal of Networks, vol. 8, no. 6, pp. 1432-1439, Jun. 2013.

See Also

findClusters

`regroupDissolved` *regroupDissolved*

Description

This function is used within the core function when the selected cluster to be dissolved has only one municipality. The call, inside this function, to the two functions determineRegroupList and determineCohesion allows to identify the new cluster to which the municipality is temporarily assigned. The output of the function is different whether the new cluster is a proper one or it is the reserve list (i.e. it does not exist a cluster to whom it can be assigned).

Usage

```
regroupDissolved(clusterData)
```

Arguments

`clusterData`	List (three components) containing all defining information on the current clusters found by the algorithm.

Note

This function is called and used internally by the main function findClusters.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo' and Guido van den Heuvel

See Also

findClusters

regroupDissolved.ncom *regroupDissolved.ncom*

Description

This function is used within the core function findcluster when the selected cluster to be dissolved has more than one municipality. The function identifies the new cluster to which the selected municipality is temporarily assigned. The output of the function is different whether the new cluster is a proper one or it is the reserve list (i.e. it does not exist a cluster to whom it can be assigned).

Usage

```
regroupDissolved.ncom(clusterData, index.com.2diss)
```

Arguments

`clusterData`	List (three components) containing all defining information on the current clusters found by the algorithm.
`index.com.2diss`	integer containing the identifier of the community inside the dissolved cluster that will be assigned to a different cluster.

Note

This function is called and used internally by the main function findClusters.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo' and Guido van den Heuvel

See Also

findClusters

Sardinia *Sardinia*

Description

data frame/data.table containing the commuting flows between municipalities in Sardinia Island (Italy) from 2001 Italian Population Census (the complete data set http://www.istat.it/it/archivio/157423). Each row corresponds to an observation i.e. a flow and each column corresponds to a variable. The variables are: community_live, community work and amount, in this order. The meaning of the variables is the following: community_live, integer, id number of the community (elementary territorial unit) where the commuter/commuters live, community_work, integer, containing the id number of the community (elementary territorial unit) where the commuter/commuters work and amount, numeric containing the number of commuter/commuters commuting between community_live and community_work (direction is important).

Usage

```
data(Sardinia)
#Encoding(names.Sardinia$com.name)="latin1" # to deal with a specific accent.
```

Author(s)

Daniela Ichim, Luisa Franconi and Michele D'Alo'

---

shpBrindisi *Polygons (shape files) information for communities in Brindisi*

Description

R object of class SpatialPolygonsDataFrame that holds polygons with attributes for each community in Brindisi provincie as from Italian Population Census 2001.

Usage

```
data("shpBrindisi")
```

Format

The format is: Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots ..@ data :'data.frame': 20 obs. of 5 variables: .. ..$ PRO_COM : int [1:20] 74001 74002 74003 74004 74005 74006 74007 74008 74009 74010 ... .$ COD_REG : int [1:20] 16 16 16 16 16 16 16 16 16 16 ... .$ COD_PRO : int [1:20] 74 74 74 74 74 74 74 74 74 74 ... .$ NOME_COM: Factor w/ 8092 levels "ABANO TERME",..: 924 1402 1885 1904 2138 2645 2716 2918 3493 4029 ... .$ POP2001 : int [1:20] 89081 14960 21370 6818 12078 8740 38667 36274 15371 27587 ... ..@ polygons :List of 20 .. ..$:Formal class 'Polygons' [package "sp"] with 5 slots .. ..@ Polygons :List of 1 .. ..$:Formal class 'Polygon' [package "sp"] with 5 slots ....@ plotOrder : int [1:20] 1 12 8 3 7 10 2 11 17 15 ... ..@ bbox : num [1:2, 1:2] 17.3 40.4 18.1 40.9 .. ..- attr(*, "dimnames")=List of 2 .. .. ..$: chr [1:2] "x" "y" .. .. ..$: chr [1:2] "min" "max" ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot .. .. ..@ projargs: chr "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"

Details

The object contains 5 slots. The data slot presents the following variables (length 20, number of communities in Brindisi NUTS3 region): PRO_COM integer containing the id code of the community; COD_REG integer: code for the region (equal to 16); COD_PRO integer: code for the provincie (NUTS3); NOME_COM: character containing the name of the community; POP2001: integer containg the population of the community as from 2001 Italian Population Census.

Source

The data is taken from the shape files (in generalised form) of the whole of Italian municipalities at the 2001 Populalation Census published by Istat: http://www.istat.it/it/archivio/124086

Examples

```
#library(data.table)
## Not run:
data(shpBrindisi)
plot(shpBrindisi)
## maybe str(shpBrindisi) ;  ...

## End(Not run)
```

---

shpSardinia *shpSardinia*

Description

R object of class SpatialPolygonsDataFrame that holds polygons with attributes for each community in Sardinia island as from Italian Population Census 2001.

Usage

```
data("shpSardinia")
```

Format

The format is: Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots ..@ data :'data.frame': 377 obs. of 5 variables: .. ..$ PRO_COM : int [1:377] 90001 90002 90003 90004 90005 90006 90007 90008 90009 90010 ... .$ COD_REG : int [1:377] 20 20 20 20 20 20 20 20 20 20 ... .$ COD_PRO : int [1:377] 90 90 90 90 90 90 90 90 90 90 ... .$ NOME_COM: chr [1:377] "AGGIUS" "ALA' DEI SARDI" "ALGHERO" "ANELA" ... .$ POP2001 : int [1:377] 1686 1949 38404 817 847 10730 677 2181 3177 501 ... ..@ polygons :List of 377 .. ..@ plotOrder : int [1:377] 64 47 191 52 274 6 254 106 3 151 ... ..@ bbox : num [1:2, 1:2] 8.13 38.86 9.83 41.27 .. ..- attr(*, "dimnames")=List of 2 .. .. ..$: chr [1:2] "x" "y" .. .. ..$: chr [1:2] "min" "max" ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot .. .. ..@ projargs: chr "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"

Details

The object contains 5 slots. The data slot presents the following variables (length 377, number of communities in Sardinia): PRO_COM integer containing the id code of the Sardinian community; COD_REG integer: code for the region Sardinia (equal to 20); COD_PRO integer: code for the provincie (NUTS3); NOME_COM: character containing the name of the community; POP2001: integer containg the population of the community as from 2001 Italian Population Census.

Source

The data is taken from the shape files (in generalised form) of the whole of Italian municipalities at the 2001 Populalation Census published by Istat: http://www.istat.it/it/archivio/124086

Examples

```
## Not run:
data(shpSardinia)
plot(shpSardinia)

## End(Not run)
## maybe str(shpSardinia) ; ...
```

`StatClusterData` *StatClusterData*

Description

This function computes several statistics on a given set of labour market areas (a given partition).

Usage

```
StatClusterData(lma,param,threshold,dat)
```

Arguments

`lma`	A list of data.table containing information on the labour market areas. Three components: clusterList, LWClus and marginals. clusterList is a data.table containing the variables community, cluster and EMP_live; everything else will be ignored. LWClus is a data.table containing the variables cluster_live, cluster_work and amount; everything else will be ignored. marginals is a data.table containing the variables cluster, amount_live and amount_work; everything else will be ignored. In each data.table object, the order of the variables is mandatory. The lma names should have not been assigned; otherwise use function DeleteLmaName.
`param`	numeric vector: the set of parameters corresponding to the lma object, i.e. minSZ,minSC,tarSZ,tarSC, respectively. See function findClusters.
`threshold`	numeric. It is used to identify particular small labour market areas or flows.
`dat`	data frame/data.table containing the commuting flows between communities (see for example Sardinia).

Value

A list with the following components:

`marginals`	data.table containing the following variables for each LMA. **LMA:** positive integer; labour market areas ID **EMP_live:** numeric; number of commuters who live in the area **EMP_work:** numeric; number of commuters working in the area **validity:** numeric; validity value computed with the current parameters

EMP_live_work: numeric; number of commuters living and working in the area

lma_commuter_percent: numeric; the quantity: (EMP_live-EMP_live_work)+(EMP_work-EMP_live_work)/(2*EMP_live_work)

Home_Work_Ratio: numeric; the quantity ((EMP_live-EMP_live_work)-(EMP_work-EMP_live_work))/EMP_live_work

SC_demand_side: numeric; demand side self-containment

SC_supply_side: numeric; supply side self-containment

N_com: integer; number of communities forming the LMA

InternalCohesionLink: numeric; consistency of internal relationships. It is given by the ratio between number of links between communities inside LMA, excluding itself, and the maximum number of possible links, i.e. (N_com * (N_com-1)). See [1].

InternalCohesionFlows: numeric; intensity of internal relationships. It is the percentage of internal flows (excluding flows having as origin and destination the same node) of the LMA between different communities w.r.t the total internal flows. See [3].

NbCentralComm: integer; number of communities having a centrality index greater than 1 (for communities with more than 100 workers, the centrality index is the ratio between net incoming flows and net outgoing flows).

N_links_in: integer; number of LMAs whose residents work in the current LMA (including itself)

N_links_out: integer; number of LMAs where the residents of the current LMA work (including itself)

`StatFlows` list containing several statistics on flows and links between the labour market areas of the given partition.

N_links: numeric; number of links between LMAs

PercNbLinksLessThreshold: numeric; percentage of links corresponding to flows below threshold

summFlows: numeric vector; summary statistics on flows

summFlowsNoItself: numeric vector; summary statistics on flows, excluding the self-flows

summLinks_in: numeric vector; summary statistics on the number of incoming flows

summLinks_out: summary statistics on the number of outgoing flows

clusterMaxNlinks_in: positive integer; the LMA ID of the cluster reaching the maximum number of incoming flows

clusterMaxNlinks_out: positive integer; the LMA ID of the cluster reaching the maximum number of outgoing flows

clusterMinNlinks_in: positive integer; the LMA ID of the cluster reaching the minimum number of incoming flows

clusterMinNlinks_out: positive integer; the LMA ID of the cluster reaching the minimum number of outgoing flows

`StatQuality` list containing several statistics on the given partition:

NbClusters: integer; number of clusters

NbClusterUniqueCom: integer; number of clusters with an unique community

NbClustersValidLess1: integer; number of clusters with validity smaller than 1

NbClustersNoCentralCom: integer; number of clusters with no communities having a centrality measure greater than 1

Mean.SC_demand_side: numeric; mean of the demand side self-containment of the clusters in the partition

Std.SC_demand_side: numeric; standard deviation of the demand side self-containment

Mean.SC_supply_side: numeric; mean of the supply side self-containment of the clusters in the partition

Std.SC_supply_side: numeric; standard deviation of the supply side self-containment

Q1.InternalCohesionFlows: numeric; first quartile of the InternalCohesionFlows

Q2.InternalCohesionFlows: numeric; median of the InternalCohesionFlows

Q3.InternalCohesionFlows: numeric; third quartile of the InternalCohesionFlows

Q1.InternalCohesionLink: numeric; first quartile of the InternalCohesionLink

Q2.InternalCohesionLink: numeric; median of the InternalCohesionLink

Q3.InternalCohesionLink: numeric; third quartile of the InternalCohesionLink

Q1.EMP_live: numeric; first quartile of the residents

Q2.EMP_live: numeric; median of the residents

Q3.EMP_live: numeric; third quartile of the residents

Mean.EMP_live: numeric; mean value of the residents

Std.EMP_live: numeric; standard deviation of the residents

Min.EMP_live: numeric; minimum value of the residents

Max.EMP_live: numeric; maximum value of the residents

Q1.EMP_work: numeric; first quartile of the workers/jobs

Q2.EMP_work: numeric; median of the workers/jobs

Q3.EMP_work: numeric; third quartile of the workers/jobs

Mean.EMP_work: numeric; mean value of the workers

Std.EMP_work: numeric; standard deviation of the workers

Min.EMP_work: numeric; minimum value of the workers

Max.EMP_work: numeric; maximum value of the workers

Q1.EMP_live_work: numeric; first quartile of the commuters living and working in the same area

Q2.EMP_live_work: numeric; median of the commuters living and working in the same area

Q3.EMP_live_work: numeric; third quartile of the the commuters living and working in the same area

Min.EMP_live_work: numeric; minimum value of the commuters living and working in the same area

Max.EMP_live_work: numeric; maximum value of the commuters living and working in the same area

Mean.lma_commuter_percent: numeric; mean value of the quantity: (EMP_live-EMP_live_work)+(EMP_work-EMP_live_work)/(2*EMP_live_work)

Std.lma_commuter_percent: numeric; standard deviation of the quantity (EMP_live-EMP_live_work)+(EMP_work-EMP_live_work)/(2*EMP_live_work)

Mean.Home_Work_Ratio: numeric; mean value of the quantity ((EMP_live-EMP_live_work)-(EMP_work-EMP_live_work))/EMP_live_work

Std.Home_Work_Ratio: numeric; standard deviation of the quantity ((EMP_live-EMP_live_work)-(EMP_work-EMP_live_work))/EMP_live_work

Q_modularity: numeric; Q_modularity index

`param:` numeric vector; it contains the parameters of the given solution, i.e. the output of the function findClusters. The parameters are minSZ,minSC,tarSZ,tarSC.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo'

References

[1] Erba, A., D'Angio', A. e Marzulli, S. (1990). Partizioni funzionali del territorio: il modello Isers, Franco Angeli, Milano.

[2] Franconi, L., D'Alo' M. and Ichim, D. (2016). Istat implementation of the algorithm to develop Labour Market Areas.

[3] Lipizzi, F. (2014). Strumenti e indicatori per la misura della consistenza e omogeneita' delle aree funzionali. XXXV Conferenza annuale AISRe, "Uscire dalla crisi. Citta', Comunita' e Specializzazione Intelligenti", Padova, 11-13 September 2014.

See Also

findClusters

`StatReserveList` *StatReserveList*

Description

This function computes several statistics on the reserve.list (see function findCluster).

Usage

```
StatReserveList(reserve.list, dat)
```

Arguments

reserve.list	list containing the information on the reserve list. It is the output of the findClusters function. Each component of the list should contain (at least): the type of assignment to the reserve list, the iteration number, the cluster dissolved to be assigned to the reserve list and its validity, the community assigned to the reserve list. See function findClusters.
dat	data.frame/data.table containing the commuting data. Each row corresponds to an observation, i.e. a flow, and each column corresponds to a variable. The variables are named: community_live: (integer) contains the id number of the elementary zone of residence, community_work: (integer) contains the id number of the working community (elementary zone of work), amount: (numeric) contains the number of employees commuting between the "community_live" and "community_work" (direction is important). Missing values (NAs) are not allowed. The community id must be strictly positive. Negative flows are not allowed.

Value

A list with the following components:

NumComm	integer; number of communities in the reserve list
NumClus	integer; as each community in the reserve list initially belonged to a cluster, this statistics reports the number of different clusters in the reserve list. The clusters ID are the ones initially generated by the algorithm.
NumUniqueClus	integer; number of clusters (for communities in the reserve list) with a unique community.
summaryCommByClus	numeric vector; summary statistics on the number of communities, by cluster. Only the communities included in the reserve list are considered.
summaryValidities	numeric vector; summary statistics on validity values of the clusters when they were considered for the reserve.list.
TypesTable	frequency table of the types of assignment of the communities to the reserve list.
Residents	data.table of two columns: communities in the reserve list and their residents.
summaryResidByComm	numeric vector; summary statistics on the number of residents by community in the reserve list.
Workers	data.table of two columns: communities in the reserve list and their jobs.
summaryWorkersByCom	numeric vector; summary statistics on the number of workers by community in the reserve list.

Author(s)

Daniela Ichim, Luisa Franconi, Michele D'Alo'

References

[1] Franconi, L., D'Alo' M. and Ichim, D. (2016). Istat implementation of the algorithm to develop Labour Market Areas.

See Also

findClusters

Index