**congruence**  
*Compute Various Measures of Functional Congruence*

**Description**
Calculates the specified congruence measure (as indicated by `type`) on the `metamatrix` object in `m`.

**Usage**
```
congruence(m, type=c("communication", "resource"))
```

**Arguments**
- `m`: An object of class `metamatrix`
- `type`: The type of congruence to be computed; options are "communication" and "resource"

**Details**
Organizational congruence indices measure the extent to which there is a close match between the requirements of task performance (as indicated by task dependencies) and the capabilities of organizational actors (as indicated by assignment/allocation of tasks, resources, etc.). As implemented here, congruence is always given as

\[
1 - \frac{d(G_R, G_C)}{\text{max } d(G_R, G_C)}
\]

where \(d\) is the Hamming distance (see `hdist`), \(G_R\) is a "requirement" structure, and \(G_C\) is a corresponding "capability" structure. Thus, congruence measures are on the \([0,1]\) interval, with 1 indicating a perfect match between requirements and capabilities, and 0 indicating a complete *disjunction* between the same.

The specific congruence measures implemented here are as follows:

1. "communication": Communication congruence assesses the extent to which interdependence between tasks is mirrored by the communication network (i.e., the extent to which those who are assigned to tasks communicate with those assigned to the tasks on which their tasks depend). For this measure, \(G_R\) is the Task\times Task graph, and \(G_C\) is the graph given by (Task\times Personnel)*(Personnel\times Task).
2. "resource": Resource congruence measures the extent to which task resource requirements match the task and resource assignment networks (i.e., to what degree does the assignment of actors to tasks and resources reflect the task resource requirements). Here, \(G_R\) is the Resource\times Task graph, and \(G_C\) is the graph given by (Resource\times Personnel)*(Personnel\times Task).

Note that the `congruence` routine will exit (generating an error) if the specific metamatrix entries it requires are not present (see `metamatrix`). The required entries are as specified above.
Value

A congruence score

Requires

sna

Note

It is important to bear in mind that lack of congruence can result not only from an inability to meet requirements, but also from slack resources (i.e., capabilities which are not required). Congruence simply measures the total agreement between capabilities and requirements.

Author(s)

Carter T. Butts (ctb@andrew.cmu.edu)

References


See Also

metamatrix, redundancy

Examples

#Randomly determine numbers of personnel, knowledge elements, #resources, tasks, and organizations

np<-sample(5:15,1)
nk<-sample(5:15,1)
nr<-sample(5:15,1)
nt<-sample(5:15,1)
nk<-sample(5:15,1)

#Define a temporary function to draw arbitrary random matrices

r<-function(r,c){
    matrix(sample(c(0,1),r*c,replace=T),ncol=c)
}

#Draw a random (symmetric) metamatrix

dat<-metamatrix(pp=r(np,nk),pk=r(np,nk),pr=r(np,nr),pt=r(np,nt),
    po=r(np,no),kkr=r(nk,nk),krr=r(nk,nr),kt=r(nk,nt),ko=r(nk,no),
    rr=r(nr,nr),t=r(nr,nt),ro=r(nr,no),tt=r(nt,nt),to=r(nt,no),
    oo=r(no,no),fill.in=TRUE,p.lab=paste("p",1:np,sep=""),
    k.lab=paste("k",1:nk,sep=""),r.lab=paste("r",1:nr,sep=""),
    ...
t.lab=paste("t",i:nt,sep=""), o.lab=paste("o",i:no,sep="")

#Compute resource congruence
congruence(dat,"resource")

metamatrix

Generate a metamatrix Object from an Input Graph Set

Description

metamatrix generates a metamatrix object from a series of input graphs, possibly using transposed inputs to fill in missing networks (if fill.in.transpose is set). Labels for personnel, knowledge, resource, task, and organizational elements are given by p.lab, k.lab, r.lab, t.lab, and o.lab, respectively.

Usage

metamatrix(mat=NULL, p.ind=NULL, k.ind=NULL, r.ind=NULL, t.ind=NULL, o.ind=NULL, pp=NULL, pk=NULL, pr=NULL, pt=NULL, po=NULL, kp=NULL, kr=NULL, kt=NULL, ko=NULL, rp=NULL, rk=NULL, rr=NULL, rt=NULL, ro=NULL, tp=NULL, tk=NULL, tp=NULL, tt=NULL, tp=NULL, op=NULL, ok=NULL, or=NULL, ot=NULL, oo=NULL, fill.in.transpose=TRUE, p.lab=NULL, k.lab=NULL, r.lab=NULL, t.lab=NULL, o.lab=NULL)

Arguments

mat A combined adjacency matrix on all metamatrix elements
p.ind The indices of mat corresponding to personnel elements (if mat!=NULL)
k.ind The indices of mat corresponding to knowledge elements (if mat!=NULL)
r.ind The indices of mat corresponding to resource elements (if mat!=NULL)
t.ind The indices of mat corresponding to task elements (if mat!=NULL)
o.ind The indices of mat corresponding to organizational elements (if mat!=NULL)
pp The PersonnelxPersonnel adjacency matrix (if mat==NULL)
pk The PersonnelxKnowledge adjacency matrix (if mat==NULL)
pr The PersonnelxResource adjacency matrix (if mat==NULL)
pt The PersonnelxTask adjacency matrix (if mat==NULL)
po The PersonnelxOrganization adjacency matrix (if mat==NULL)
kp The KnowledgexPersonnel adjacency matrix (if mat==NULL)
kk The KnowledgexKnowledge adjacency matrix (if mat==NULL)
kr The KnowledgexResource adjacency matrix (if mat==NULL)
kt The KnowledgexTask adjacency matrix (if mat==NULL)
ko The KnowledgexOrganization adjacency matrix (if mat==NULL)
rp The ResourcexPersonnel adjacency matrix (if mat==NULL)
The ResourcexKnowledge adjacency matrix (if mat==NULL)
The ResourcexResource adjacency matrix (if mat==NULL)
The ResourcexTask adjacency matrix (if mat==NULL)
The ResourcexOrganization adjacency matrix (if mat==NULL)
The TaskxPersonnel adjacency matrix (if mat==NULL)
The TaskxKnowledge adjacency matrix (if mat==NULL)
The TaskxResource adjacency matrix (if mat==NULL)
The TaskxTask adjacency matrix (if mat==NULL)
The TaskxOrganization adjacency matrix (if mat==NULL)
The OrganizationsxPersonnel adjacency matrix (if mat==NULL)
The OrganizationsxKnowledge adjacency matrix (if mat==NULL)
The OrganizationsxResource adjacency matrix (if mat==NULL)
The OrganizationsxTask adjacency matrix (if mat==NULL)
The OrganizationsxOrganization adjacency matrix (if mat==NULL)

Should metamatix attempt to use other-triangular cells (transposed) to fill in missing (NULL) metamatix cells?

Labels for personnel elements
Labels for knowledge elements
Labels for resource elements
Labels for task elements
Labels for organizational elements

The metamatrix representation of organizational structure seeks to summarize organizations via a set of interrelated networks defined on personnel, knowledge, resource, task, and (external) organizational elements. The "metamatix" for which the representation is named is formed by the 25 dyadic compositions of these five basic elements, to wit:

```
P  K  R  T  O
P  PP PK PR PT PO
K  KP KK KR KT KO
R  RP RK RR RT RO
T  TP TK TR TT TO
O  OP OK OR OT OO
```

The composition operation here can generally be interpreted as "is assigned to", "can communicate with", "requires", "controls", etc., depending on the composition in question; for most purposes, a more useful (albeit abstract) interpretation is that a "path of action" links the elements on one side of the relation with those on the other. Thus, the KP and PT networks can be interpreted as the assignment of knowledge to personnel and the assignment of personnel to tasks, respectively, with their (inner) product reflecting the network...
of paths from knowledge to tasks via personnel. By means of operations such as multiplication and transposition, it is possible to construct a fairly elaborate algebra of organizational structure. Expressions in this algebra can then be used to discover (or predict) properties of organizational structure.

Data regarding relations among metamatrix elements may be supplied in two ways. First, a combined adjacency matrix for all elements may be given via the mat parameter, with the indices corresponding to personnel, knowledge, resource, task, and organizational elements given by p.ind, k.ind, r.ind, t.ind, and o.ind, respectively. (In this case, the individual metamatrix cell parameters are ignored, although the labeling arguments function normally.) Alternately, the adjacency matrices for particular metamatrix cells may be supplied directly, using the appropriate parameter (i.e., pp for Personnel \times Personnel, kt for Knowledge \times Task, etc.). In this case, mat must be set to NULL and the index parameters are ignored.

Practically speaking, all 25 relations of the metamatrix are rarely used in any one analysis; generally, only a subset will be desired (not to mention available). Given this, the user may elect not to provide some of the basic relations (which will then be coded NULL). In many contexts, many if not all of the lower triangular entries of the metamatrix will be equal to the transposes of their upper triangular counterparts (or vice versa), e.g., when there is no substantive distinction between the assignment of knowledge to personnel and the assignment of personnel to knowledge. In such cases, the argument fill.in.transpose may be used to attempt to fill in any unprovided metamatrix spells with the corresponding entry on the opposite triangle (if present). Labels for organizational elements (personnel, tasks, etc.) are provided by the indicated arguments, and will be used to label all metamatrix entries as appropriate.

Value

An object of class metamatrix, containing:

- meta   A list of adjacency matrices
- np    Number of personnel elements
- nk    Number of knowledge elements
- nr    Number of resource elements
- nt    Number of task elements
- no    Number of organizational elements
- p.lab Personnel element labels
- k.lab Knowledge element labels
- r.lab Resource element labels
- t.lab Task element labels
- o.lab Organizational element labels

Warning

All graphs, labels, etc. must come from the same organization; i.e., the number of each element must remain constant across structures, and elements must always be given in the same order. Violating either assumption will result in error messages and/or garbled data.
Author(s)

Carter T. Butts (ctb@andrew.cmu.edu)

References


See Also

plot.metamatrix, print.metamatrix, read.metamatrix, nos

Examples

#Randomly determine numbers of personnel, knowledge elements, #resources, tasks, and organizations
np<sample(5:15,1)
nk<sample(5:15,1)
r<sample(5:15,1)
nt<sample(5:15,1)
no<sample(5:15,1)

#Define a temporary function to draw arbitrary random matrices
r<-function(r,c){
  matrix(sample(c(0,1),r*c,replace=T),nrow=r,ncol=c)
}

#Draw a random (symmetric) metamatrix
data<-metamatrix(pp=r(np, np), pk=r(np, nk), pr=r(np, nr), pt=r(np, nt),
  po=r(np, no), ke=r(nk, nk), kr=r(nk, nr), kt=r(nk, nt), ko=r(nk, no),
  or=r(nr, nr), ot=r(nr, nt), oo=r(nr, no), fill.in = transpose = TRUE, p.lab = paste("p", 1:np, sep=""),
  k.lab = paste("k", 1:nk, sep=""), r.lab = paste("r", 1:nr, sep=""),
  t.lab = paste("t", 1:nt, sep=""), o.lab = paste("o", 1:no, sep=""))

#Display some basic information
data

#Now repeat, using an aggregated data matrix
np.i<1:np
nk.i<-(np+1):(np+nk)
r.i<-(np+nk+1):(np+nk+nr)
nt.i<-(np+nk+nr+1):(np+nk+nr+nt)
no.i<-(np+nk+nr+nt+1):(np+nk+nr+nt+no)
mat<-rgraph(sum(np, nk, nr, nt, no))
data<-metamatrix(mat, np.i, nk.i, nr.i, nt.i, no.i,
  p.lab = paste("p", 1:np, sep=""), k.lab = paste("k", 1:nk, sep=""),
r.lab=paste("r",i:nr,sep=""),t.lab=paste("t",i:nt,sep=""),
o.lab=paste("o",i:no,sep="")

#See again the basic information
dat

#Plot the metamatrx
plot(dat)

plot.metamatrix  
Plotting for metamatrix Objects

Description
Plots all available cells of a given metamatrix object (using gplot) in a pairs-like display.

Usage
plot.metamatrix(m, main=NULL, mode="mds", ...)

Arguments
m  An object of class metamatrix
main  Main title text (optional)
mode  Plotting mode to be used by gplot
...  Additional arguments to gplot

Details
plot.metamatrix shows all available metamatrix graphs, using gplot to plot each; where an entry is missing, the display is left blank. The mode argument may be used to set the graph layout mode, as per the gplot instructions. This can be useful for getting a feel for the overall organizational structure, and can be used to target particular graphs for further analysis (a la pairs).

Value
None

Note
This routine uses something of a hack on gplot, since the latter does not deal with two-mode matrices by default. Eventually, this capability will be natively available.

Author(s)
Carter T. Butts (ctb@andrew.cmu.edu)
References


See Also

plot, gplot, metamatrix

Examples

#Randomly determine numbers of personnel, knowledge elements, #resources, tasks, and organizations
np<-sample(b:15,1)
nk<-sample(b:15,1)
nr<-sample(b:15,1)
nr<-sample(b:15,1)
no<-sample(b:15,1)

#Define a temporary function to draw arbitrary random matrices
r<-function(r,c){
  matrix(sample(c(0,1),r*c,replace=T),nrow=r,ncol=c)
}

#Draw a random (symmetric) metamatrix
dat<-metamatrix(p=r(np,np),pk=r(np,nk),pr=r(np,nr),pt=r(np,nt),
  po=r(np,po),ko=r(np,nk),kr=r(nk,nr),kt=r(nk,nt),ko=r(nk,no),
  rr=r(nr,nr),rt=r(nr,nt),ro=r(nr,po),tt=r(nt,nt),to=r(nt,nt),
  oo=r(no,oo),fill.in.transpose=TRUE,p.lab=paste("p",1:np,sep=""),
  k.lab=paste("k",1:nk,sep=""),r.lab=paste("r",1:nr,sep=""),
  t.lab=paste("t",1:nt,sep=""),o.lab=paste("o",1:no,sep=""))

#Plot the metamatrix
plot(dat)

print.mematix Printing for metamatrix Objects

Description

Prints simple descriptives for a metamatrix object.

Usage

print.mematix(m)
Arguments

\( m \)  
An object of class \texttt{metamatrix}

Value

None

References


See Also

\texttt{metamatrix}

Examples

#Randomly determine numbers of personnel, knowledge elements,  
#resources, tasks, and organizations
np<-sample(5:15,1)  
nk<-sample(5:15,1)  
nr<-sample(5:15,1)  
nr<-sample(5:15,1)  
nc<-sample(5:15,1)

#Define a temporary function to draw arbitrary random matrices
r<-function(r,c){
    matrix(sample(c(0,1),r*c,replace=T),nrow=r,ncol=c)
}

#Draw a random (symmetric) metamatrix
dat<-metamatrix(pp=r(np,np),pk=r(np,nk),pr=r(np,nr),pt=r(np,nt),  
pq=r(np,no),kk=r(nk,nk),kr=r(nk,nr),kt=r(nk,nt),ko=r(nk,no),  
r=1(nr,nr),rt=r(nr,nt),ro=r(nr,no),tt=r(nt,nt),to=r(nt,no),  
oo=r(no,no),fill.in,transpose=TRUE,p.lab=paste("p","i:np,sep="")  
,k.label=paste("k","i:nk,sep="",r.label=paste("r","i:nr,sep="",  
t.label=paste("t","i:nt,sep="",o.label=paste("o","i:no,sep=""))  

#Display some basic information

dat
**read.metamatrix.nos**  
*Read metamatrix Data from a (N)ew-(O)rg(S)lat File*

**Description**

Reads the input file given in file (in NOS format), processing the result using `metamatrix`.

**Usage**

```r
read.metamatrix.nos(file, fill.in.transpose=TRUE, p.lab=NULL, k.lab=NULL, r.lab=NULL, t.lab=NULL, o.lab=NULL)
```

**Arguments**

- `file`  
The file to be imported
- `fill.in.transpose`  
  Should `metamatrix` attempt to use other-triangular cells (transposed) to fill in missing (NULL) metamatrix cells?
- `p.lab`  
  Labels for personnel elements
- `k.lab`  
  Labels for knowledge elements
- `r.lab`  
  Labels for resource elements
- `t.lab`  
  Labels for task elements
- `o.lab`  
  Labels for organizational elements

**Details**

`read.metamatrix.nos` reads a NOS-format input file (see `read.nos`) containing a combined metamatrix adjacency matrix (see `metamatrix`) and uses `metamatrix` to process the result. Metamatrix elements are identified by their row color in the NOS header, as follows:

- 0: Personnel element
- 1: Knowledge element
- 2: Resource element
- 3: Task element
- 4: Organizational element

Other command parameters have the same behavior here as they do in `metamatrix`, save in that default labels are generated where none are specified by the user.

**Value**

An object of class `metamatrix`

**Note**

This is probably the easiest way to read `metamatrix` data into R at present.
Author(s)
Carter T. Butts (ctb@andrew.cmu.edu)

References


See Also
metamatrix, read.mos, scan

Examples

<table>
<thead>
<tr>
<th>redundancy</th>
<th>Compute an Index of Redundancy in Assignments</th>
</tr>
</thead>
</table>

Description

redundancy computes either row or column redundancy scores (as per type) on the adjacency matrix in mat.

Usage

redundancy(mat, type=c("row", "column"))

Arguments

mat A single adjacency matrix
type One of "row" or "column"

Details

The redundancy of a given row (i) in adjacency matrix A is defined by $r_r (i) = \left( \sum_{j=1}^{n} A_{ij} \right) - 1$; the redundancy of a corresponding column is similarly defined by $r_c (i) = \left( \sum_{j=1}^{n} A_{ji} \right) - 1$. For an entire adjacency matrix, then, the normalized row redundancy is equal to

$$R_r (A) = \frac{\sum_{i=1}^{n} r_r (i)}{n(m - 1)}$$
with normalized column redundancy given by
\[
R_c(A) = \frac{\sum_{i=1}^{m} r_c(i)}{m(n-1)}
\]

It is these last scores which are computed by **redundancy**. Loosely speaking, the redundancy of a particular graph can be interpreted as an index of the extent to which it has more edges which are needed to make it row or column covered (depending on the type of redundancy). Such a measure might be relevant for assessing overall coverage in assignment networks, where task completion may be threatened if at least one person is not assigned to each task/resource/etc.

**Value**

The redundancy score

**Note**

Be sure to send this routine a single adjacency matrix, rather than a **metamatrix** object.

**Author(s)**

Carter T. Butts (ctb@andrew.cmu.edu)

**References**


**See Also**

congruence

**Examples**

```r
#Generate a random graph, and find the row redundancy
redundancy(rgraph(10), "row")
```
usage

Compute Predicted Usage Matrices

Description

Computes the predicted usage matrix indicated by type for metamatrix object m.

Usage

usage(m, type=c("resource", "skill"))

Arguments

m An object of class metamatrix
type The type of usage to assess; must be one of "resource" or "skill"

Details

A predicted usage matrix is always a matrix of the form

\[ U = A_R - A_C \]

where \( A_R \) is a matrix of requirements, and \( A_C \) is a matrix of capabilities. The entries of \( U \) thus provide a rough idea of the extent to which any given requirement is met by organizational capabilities: positive entries indicate that insufficient capabilities are present to meet the requirements in question, while negative entries indicate requirements for which some capabilities will be underutilized (i.e., areas in which there are slack resources). The particular usage matrices computed by usage are:

1. "skill": The skill usage matrix. For this analysis, \( A_R \) is the KnowledgeTask matrix, and \( A_C \) is given by (KnowledgePersonnel)*(PersonnelTask).
2. "resource": The resource usage matrix. For this analysis, \( A_R \) is the ResourceTask matrix, and \( A_C \) is given by (ResourcePersonnel)*(PersonnelTask).

Value

The predicted usage matrix

Author(s)

Carter T. Butts (ctb@andrew.cmu.edu)

References


See Also

congruence

Examples

# Randomly determine numbers of personnel, knowledge elements, 
# resources, tasks, and organizations
np <- sample(5:15,1)
nk <- sample(5:15,1)
nr <- sample(5:15,1)
nt <- sample(5:15,1)
no <- sample(5:15,1)

# Define a temporary function to draw arbitrary random matrices
r <- function(r,c){
  matrix(sample(c(0,1),r*c,replace=T),nrow=r,ncol=c)
}

# Draw a random (symmetric) metamatrix
dat <- metamatrix(pp=r(np,np),pk=r(np,nk),pr=r(np,nr),pt=r(np,nt),
  po=r(np,no),kk=r(nk,nk),kr=r(nk,nr),kt=r(nk,nt),ko=r(nk,no),
  rr=r(nr,nr),rt=r(nr,nt),ro=r(nr,no),tt=r(nt,nt),to=r(nt,no),
  oo=r(no,no),fill.in.transpose=TRUE,p.lab=paste("p",1:np,sep=""),
  k.lab=paste("k",1:nk,sep=""),r.lab=paste("r",1:nr,sep=""),
  t.lab=paste("t",1:nt,sep=""),o.lab=paste("o",1:no,sep=""))

# Calculate predicted skill usage
usage(dat,"skill")


workload  Calculate Various Measures of Predicted Workload

Description

Calculates the workload measure given by type on the metamatrix object in m.

Usage

workload(m, type=c("potential", "actual"))

Arguments

m  An object of class metamatrix

type  The type of workload measure to compute; must be one of "potential" or "actual"
Details

Workload measures attempt to predict (at least roughly) the extent to which a given actor will have the ability and/or requirement of completing numerous tasks. Here, we consider two:

1. "potential": The vector of potential workload scores is given by $A_{pk}A_{kt}A_{lp}$, and the ith entry corresponds to the number of skills that the ith actor can contribute to all tasks (with the same skill counting multiple times if applicable for multiple tasks). This is in some sense a measure of each actor’s individual ability to contribute to task completion, resource and task assignments notwithstanding.

2. "actual": The vector of "actual” workload scores (something of a misnomer) is given by the diagonal entries of $A_{pk}A_{kt}A_{lp}$. This measure is probably best interpreted as the quantity of each actor’s potential workload which accrues to tasks to which he or she is actually assigned. Obviously, "actual" workload is always less than or equal to potential workload, and serious discrepancies between the two may indicate inefficiencies in task assignment.

Obviously, other, more sophisticated measures of workload are possible. Some of these will doubtless be added, eventually.

Value

A vector of workload measures

Author(s)

Carter T. Butts (ctb@andrew.cmu.edu)

References


See Also

usage

Examples

#Randomly determine numbers of personnel, knowledge elements, #resources, tasks, and organizations
mp<-sample(5:15,1)
mk<-sample(5:15,1)
mr<-sample(5:15,1)
mt<-sample(5:15,1)
no<-sample(5:15,1)
#Define a temporary function to draw arbitrary random matrices
r<-function(r,c){
    matrix(sample(c(0,1),r*c,replace=T),nrow=r,ncol=c)
}

#Draw a random (symmetric) metamatrix
dat<-metamatrix(pp=r(np,np),pk=r(np,nk),pr=r(np,nr),pt=r(np,nt),
    po=r(np,no),kk=r(nk,nk),kr=r(nk,nr),kt=r(nk,nt),ko=r(nk,no),
    rr=r(nr,nr),rt=r(nr,nt),ro=r(nr,no),tt=r(nt,nt),to=r(nt,no),
    oo=r(no,no),fill.in.transpose=TRUE,p.lab=paste("p",1:np,sep=""),
    k.lab=paste("k",1:nk,sep=""),r.lab=paste("r",1:nr,sep=""),
    t.lab=paste("t",1:nt,sep=""),o.lab=paste("o",1:no,sep=""))

#Calculate potential workload
workload(dat,"potential")

#Calculate differences between "actual" and potential workload
workload(dat,"potential")-workload(dat,"actual")