Package ‘NetworkRiskMeasures’

October 12, 2022

Type Package
Title Risk Measures for (Financial) Networks
Version 0.1.4
Author Carlos Cinelli <carloscinelli@hotmail.com>, Thiago Cristiano Silva <thiagochris@gmail.com>
Maintainer Carlos Cinelli <carloscinelli@hotmail.com>
Description Implements some risk measures for (financial) networks, such as DebtRank, Impact Susceptibility, Impact Diffusion and Impact Fluidity.
License GPL-3
LazyData TRUE
Suggests testthat, igraph, covr
Depends Matrix
Imports expm, ggplot2, dplyr
URL https://github.com/carloscinelli/NetworkRiskMeasures
BugReports https://github.com/carloscinelli/NetworkRiskMeasures/issues
RoxygenNote 7.0.2
NeedsCompilation no
Repository CRAN
Date/Publication 2020-03-05 11:20:02 UTC

R topics documented:

communicability_matrix ........................................... 2
contagion ..................................................................... 3
criticality ................................................................. 5
impact_susceptibility ................................................... 6
matrix_estimation ......................................................... 8
risk_matrix ................................................................ 10
sim_data .................................................................... 12

Index 14
communicability_matrix

Computes the communicability matrix

Description

The communicability of an adjacency matrix $M$ is defined as $\exp(M)$ where $M[i,j]$ can be interpreted as the weighted sums of paths from $i$ to $j$. Recall that $\exp(M)$ can be cast into a Taylor series expansion with an infinite number additive terms. The function permits the evaluation of $\exp(M)$ using the `expm` package or using a simpler mathematical approximation. In the second case, the function truncates the infinite series by simply calculating the summation terms up to a pre-defined number of factors.

Usage

`communicability_matrix(x, terms = Inf, sparse = TRUE)`

Arguments

- `x`: a square matrix or an igraph object.
- `terms`: truncates the communicability matrix evaluation up to a pre-defined number of terms. If `terms = Inf` the function computes the matrix exponential using `expm`.
- `sparse`: should the function use sparse matrices when computing the communicability? However, if `terms = Inf` the function will use `expm` which uses dgeMatrix-class.

Value

The function returns the communicability matrix.

References


Examples

```r
# Creating example data
## Assets Matrix (bilateral exposures)
assets_matrix <- matrix(c(0, 10, 3, 1, 0, 2, 0, 3, 0), ncol = 3)
rownames(assets_matrix) <- colnames(assets_matrix) <- letters[1:3]

## Capital Buffer
buffer <- c(a = 2, b = 5, c = 2)

# Computing vulnerability
v <- vulnerability_matrix(assets_matrix, buffer, binary = TRUE)

# Computing communicability of the vulnerability matrix
communicability_matrix(v)
```
Description

Given a matrix of exposures, a vector of buffers and weights (optional) the functions simulates contagion for all the shock vectors provided. You may choose from the implemented propagation contagion method or create you own propagation method. Details on how to create your own method will be provided in a future version.

Usage

contagion(exposures,  
  buffer,  
  shock = "all",  
  weights = NULL,  
  method = c("debtrank", "threshold"),  
  ...,  
  exposure_type = c("assets", "liabilities", "impact", "vulnerability"),  
  keep.history = FALSE,  
  abs.tol = .Machine$double.eps ^ 0.2,  
  max.it = min(1000, nrow(v)*10),  
  verbose = TRUE)

Arguments

exposures an adjacency matrix, (sparse) Matrix or an igraph object with the network of bilateral exposures between vertices. By default, the function expects the exposures in the form of an assets matrix in which A -> B means that A has an asset with B. However, you can change that with the parameter exposure_type. When using a matrix, preferably it should have rows and columns names.

buffer a numeric vector with the capital buffer for each vertex. Values should be in the same row/column order as the network of bilateral exposures. The buffer is not needed if exposure_type = "vulnerability".

shock a list with the shock vectors. If "all" (the default) the function will run a simulation for the default of each vertex in the network.

weights default is NULL. You can use a numeric vector of weights to give some economic significance to the measures, like, for instance, the total assets of the nodes.

method the contagion propagation method. Currently, you should use either "debtrank" for the DebtRank propagation method or "threshold" for the traditional default cascades. The DebtRank version implemented is the one proposed in Bardoscia et al (2015). If you want to use the old "single-hit" DebtRank of Battiston et al (2012), simply provide the argument single.hit = TRUE.

... other arguments to be passed to the contagion propagation method.
exposure_type character vector indicating the type of the bilateral exposures. It can be an "assets" network (where A -> B means that A has an asset with B), a "liabilities" network (where A -> B means that A has a debt with B), a (binary) "impact" matrix (where A -> B indicates the relative impact of A in B’s capital buffer), or a (binary) "vulnerability" matrix (where A -> B indicates the relative impact A suffers from B’s default). The default is "assets".

keep.history keep all the history of stress levels? This can use a lot of memory, so the default is FALSE.

abs.tol the desired accuracy.

max.it the maximum number of iterations.

verbose gives verbose output. Default is TRUE.

Value

The function returns an object of class "contagion" with the results of the simulation.

References


Examples

# Loads simulated banking data
data("sim_data")
head(sim_data)

# seed for reproducibility
set.seed(15)

# minimum density estimation
# verbose = F to prevent printing
md_mat <- matrix_estimation(sim_data$assets, sim_data$liabilities, method = "md", verbose = FALSE)

# rownames and colnames for the matrix
rownames(md_mat) <- colnames(md_mat) <- sim_data$bank

# DebtRank simulation
contdr <- contagion(exposures = md_mat, buffer = sim_data$buffer, weights = sim_data$weights, shock = "all", method = "debtrank", verbose = FALSE)
summary(contdr)
plot(contdr)

# Traditional default cascades simulation
contthr <- contagion(exposures = md_mat, buffer = sim_data$buffer, weights = sim_data$weights, shock = "all", method = "threshold", verbose = FALSE)
summary(contthr)
# simulating shock scenarios 1% to 25% shock in all vertices
s <- seq(0.01, 0.25, by = 0.01)
shocks <- lapply(s, function(x) rep(x, nrow(md_mat))
names(shocks) <- paste(s*100, "pct shock")

cont <- contagion(exposures = md_mat, buffer = sim_data$buffer, shock = shocks,
                   weights = sim_data$weights, method = "debtrank", verbose = FALSE)
summary(cont)
plot(cont)

---

### criticality

**Criticality of the vertices**

**Description**

The criticality of a vertex measures its impact on its neighbors if it defaults. It is basically the rowSums of the impact_matrix.

**Usage**

```r
criticality(
  exposures,
  buffer,
  binary = FALSE,
  exposure_type = c("assets", "liabilities", "impact", "vulnerability")
)
```

**Arguments**

- `exposures`: an adjacency matrix, (sparse) Matrix or an igraph object with the network of bilateral exposures between vertices. By default, the function expects the exposures in the form of an assets matrix in which A -> B means that A has an asset with B. However, you can change that with the parameter exposure_type. When using a matrix, preferably it should have rows and columns names.
- `buffer`: a numeric vector with the capital buffer for each vertex. Values should be in the same row/column order as the network of bilateral exposures. The buffer is not needed if exposure_type = "vulnerability".
- `binary`: if binary = TRUE the function computes a 'binary' impact or vulnerability matrix. It truncates all values less than 1 to 0 and all values greater than 1 to 1.
- `exposure_type`: character vector indicating the type of the bilateral exposures. It can be an "assets" network (where A -> B means that A has an asset with B), a "liabilities" network (where A -> B means that A has a debt with B), a (binary) "impact" matrix (where A -> B indicates the relative impact of A in B's capital buffer), or a (binary) "vulnerability" matrix (where A -> B indicates the relative impact A suffers from B's default). The default is "assets".
Value

The function returns a (named) vector with the criticality for each vertex.

Examples

```r
# Creating example data
## Assets Matrix (bilateral exposures)
assets_matrix <- matrix(c(0, 10, 3, 1, 0, 2, 0, 3, 0), ncol = 3)
rownames(assets_matrix) <- colnames(assets_matrix) <- letters[1:3]

## Capital Buffer
buffer <- c(a = 2, b = 5, c = 2)

# Criticality
criticality(assets_matrix, buffer)
```

Description

The `impact_susceptibility` measures the feasible contagion paths that can reach a vertex in relation to its direct contagion paths. When the impact susceptibility is greater than 1, it means that the vertex is vulnerable to other vertices beyond its direct neighbors (remotely vulnerable).

The impact fluidity is simply the average of the impact susceptibility in the network.

The impact diffusion tries to capture the influence exercised by a node on the propagation of impacts in the network. The impact diffusion of a vertex is measured by the change it causes on the impact susceptibility of other vertices when its power to propagate contagion is removed from the network.

All these measures are based on the communicability of the vulnerability matrix (see `vulnerability_matrix` and `communicability_matrix`).

Usage

```r
impact_susceptibility(
exposures,
buffer,
weights = NULL,
terms = Inf,
sparse = TRUE,
binary = TRUE,
exposure_type = c("assets", "liabilities", "impact", "vulnerability")
)

impact_fluidity(
```

```r
impact_diffusion(
```
impact_susceptibility

exposures,
buffer,
weights = NULL,
terms = Inf,
sparse = TRUE,
binary = TRUE,
exposure_type = c("assets", "liabilities", "impact", "vulnerability")
)

impact_diffusion(
exposures,
buffer,
weights = NULL,
terms = Inf,
sparse = TRUE,
binary = TRUE,
exposure_type = c("assets", "liabilities", "impact", "vulnerability")
)

Arguments

exposures an adjacency matrix, (sparse) Matrix or an igraph object with the network of bilateral exposures between vertices. By default, the function expects the exposures in the form of an assets matrix in which A -> B means that A has an asset with B. However, you can change that with the parameter exposure_type. When using a matrix, preferably it should have rows and columns names.

buffer a numeric vector with the capital buffer for each vertex. Values should be in the same row/column order as the network of bilateral exposures. The buffer is not needed if exposure_type = "vulnerability".

weights default is NULL. You can use a numeric vector of weights to give some economic significance to the measures, like, for instance, the total assets of the nodes.

terms truncates the communicability matrix evaluation up to a pre-defined number of terms. If terms = Inf the function computes the matrix exponential using expm.

sparse should the function use sparse matrices when computing the communicability? However, if terms = Inf the function will use expm which uses dgeMatrix-class.

binary if binary = TRUE the function computes a 'binary' impact or vulnerability matrix. It truncates all values less than 1 to 0 and all values greater than 1 to 1.

exposure_type character vector indicating the type of the bilateral exposures. It can be an "assets" network (where A -> B means that A has an asset with B), a "liabilities" network (where A -> B means that A has a debt with B), a (binary) "impact" matrix (where A -> B indicates the relative impact of A in B's capital buffer), or a (binary) "vulnerability" matrix (where A -> B indicates the relative impact A suffers from B's default). The default is "assets".

Value

The impact_susceptibility function returns a vector with the (weighted) impact susceptibility
The `impact_fluidity` function returns a vector with the (weighted) impact fluidity of the network. The `impact_diffusion` function returns a data.frame with the vertex name and the (weighted) start, intermediate and total impact diffusion.

References


Examples

```r
# Creating example data
## Assets Matrix (bilateral exposures)
assets_matrix <- matrix(c(0, 10, 3, 1, 0, 2, 0, 3, 0), ncol = 3)
rownames(assets_matrix) <- colnames(assets_matrix) <- letters[1:3]

## Capital Buffer
buffer <- c(a = 2, b = 5, c = 2)

# Measures
impact_susceptibility(assets_matrix, buffer)
impact_fluidity(assets_matrix, buffer)
impact_diffusion(assets_matrix, buffer)
```

matrix_estimation

Matrix Estimation

Description

Methods for estimating matrix entries from the marginals (row and column sums). There are currently two methods implemented: Maximum Entropy (Upper 2004) and Minimum Density (Anand et al. 2015).

You may use the `matrix_estimation()` function, setting the desired method. Or you may use directly the `max_ent()` function for maximum entropy estimation or the `min_dens()` function for minimum density estimation.

Usage

```r
matrix_estimation(
  rowsums,
  colsums,
  method = c("me", "md"),
  ...
)
max.it = 1e+05,
```
matrix_estimation

    abs.tol = 0.001,
    verbose = TRUE
  )
  max_ent(rowsums, colsums, max.it = 1e+05, abs.tol = 0.001, verbose = TRUE)

min_dens(
  rowsums, 
  colsums, 
  c = 1, 
  lambda = 1, 
  k = 100, 
  alpha = 1/sum(rowsums), 
  delta = 1/sum(rowsums), 
  theta = 1, 
  remove.prob = 0.01, 
  max.it = 1e+05, 
  abs.tol = 0.001, 
  verbose = TRUE 
)

Arguments

  rowsums       a numeric vector with the row sums.
  colsums       a numeric vector with the column sums.
  method        the matrix estimation method. Choose "me" for maximum entropy or "md" for minimum density.
  ...           further arguments passed to or from other methods.
  max.it        the maximum number of iterations.
  abs.tol       the desired accuracy.
  verbose       gives verbose output. Default is TRUE.
  c             the 'cost' an extra link for the minimum density estimation. See Anand et al. (2015).
  lambda        you should use lambda together with k. For the first k rounds of the algorithm, the function will allocate a fraction lambda of the total, thus obtaining a "low density" solution, instead of a "minimum density" solution. See Anand et al. (2015).
  k             you should use lambda together with k. For the first k rounds of the algorithm, the function will allocate a fraction lambda of the total, thus obtaining a "low density" solution, instead of a "minimum density" solution. See Anand et al. (2015).
  alpha         weights for the row sums deviations. See Anand et al. (2015).
  delta         weights for the column sums deviations. See Anand et al. (2015).
  theta         scaling parameter. Emphasizes the weight placed on finding solutions with similar characteristics to the prior matrix. See Anand et al. (2015).
risk_matrix

Computes the (binary) impact or vulnerability matrices

Description

The function computes an impact or vulnerability matrix given a network of bilateral exposures and a vector of capital buffers.

Usage

```r
risk_matrix(
  exposures,
  buffer,
  binary = FALSE,
  exposure_type = c("assets", "liabilities", "impact", "vulnerability"),
  returns = c("impact", "vulnerability")
)```
vulnerability_matrix(
exposures,
buffer,
binary = FALSE,
exposure_type = c("assets", "liabilities", "impact", "vulnerability")
)

impact_matrix(
exposures,
buffer,
binary = FALSE,
exposure_type = c("assets", "liabilities", "impact", "vulnerability")
)

Arguments

exposures an adjacency matrix, (sparse) Matrix or an igraph object with the network
of bilateral exposures between vertices. By default, the function expects the
exposures in the form of an assets matrix in which A -> B means that A has an
asset with B. However, you can change that with the parameter exposure_type.
When using a matrix, preferably it should have rows and columns names.

buffer a numeric vector with the capital buffer for each vertex. Values should be in the
same row/column order as the network of bilateral exposures. The buffer is not
needed if exposure_type = "vulnerability".

binary if binary = TRUE the function computes a 'binary' impact or vulnerability ma-
trix. It truncates all values less than 1 to 0 and all values greater than 1 to 1.

exposure_type character vector indicating the type of the bilateral exposures. It can be an
"assets" network (where A -> B means that A has an asset with B), a "liabilities"
network (where A -> B means that A has a debt with B), a (binary) "impact"
matrix (where A -> B indicates the relative impact of A in B’s capital buffer), or
a (binary) "vulnerability" matrix (where A -> B indicates the relative impact
A suffers from B’s default). The default is "assets".

returns will the function return the impact or the vulnerability matrix? The default is
"impact".

Details

The impact matrix represents how much a vertex impacts the capital buffer of another vertex when
it defaults.

The vulnerability matrix is just the transpose of the impact matrix. It represents how much a vertex
is impacted by the default of another vertex.

Value

The function returns a (binary) impact or vulnerability matrix.
The term $V_{i,j}$ of the impact matrix represents the impact of $i$’s default in $j$’s capital buffer. The term $V_{i,j}$ of the vulnerability matrix represents how much $i$’s capital buffer is impacted by $j$’s default.

If binary = TRUE the values less than 1 are truncated to zero.

Examples

```r
# Creating example data
## Assets Matrix (bilateral exposures)
assets_matrix <- matrix(c(0, 10, 3, 1, 0, 2, 0, 3, 0), ncol = 3)
rownames(assets_matrix) <- colnames(assets_matrix) <- letters[1:3]

## Capital Buffer
buffer <- c(a = 2, b = 5, c = 2)

# Vulnerability matrices
vulnerability_matrix(assets_matrix, buffer, binary = FALSE)
vulnerability_matrix(assets_matrix, buffer, binary = TRUE)
```

---

**sim_data**  
*Simulated Interbank Data*

**Description**

A simulated dataset with interbank assets, liabilities, capital buffer and weights for 125 “banks”.
The code to generate the data is on the examples.

**Format**

A data frame with 125 rows and 5 variables

**Examples**

```r
# Simulated data for illustration purposes

# Setting Seed
set.seed(1100)

# Heavy tailed assets
assets <- rlnorm(125, 0, 2)
assets[assets < 4] <- runif(length(assets[assets < 4]))

# Heavy tailed liabilities
liabilities <- rlnorm(125, 0, 2)
liabilities[liabilities < 4] <- runif(length(liabilities[liabilities < 4]))

# Making sure assets = liabilities
assets <- sum(liabilities) * (assets/sum(assets))
```
# Buffer as a function of assets
buffer <- pmax(0.01, runif(length(liabilities)) * liabilities + abs(rnorm(125, 4, 2.6)))

# Weights as a function of assets, buffer and liabilities
weights <- (assets + liabilities + buffer + 1) + rlnorm(125, 0, 1)

# creating data.frame
sim_data <- data.frame(bank = paste0("b", 1:125), 
                        assets = assets, 
                        liabilities = liabilities, 
                        buffer = buffer, 
                        weights = weights)
Index

* dataset
  sim_data, 12

communicability_matrix, 2, 6
contagion, 3
criticality, 5
data.frame, 8
expm, 2, 7
igraph, 2, 3, 5, 7, 11
impact_diffusion
  (impact_susceptibility), 6
impact_fluidity
  (impact_susceptibility), 6
impact_matrix, 5
impact_matrix(risk_matrix), 10
impact_susceptibility, 6

Matrix, 3, 5, 7, 11
matrix, 2, 3, 5, 7, 11
matrix_estimation, 8
max_ent(matrix_estimation), 8
min_dens(matrix_estimation), 8

risk_matrix, 10
rowSums, 5

sim_data, 12

vulnerability_matrix, 6
vulnerability_matrix(risk_matrix), 10