Contents

1. Introduction 7

1.1. Pairwise Comparisons method 7

1.2. The aim of the work 8

1.3. Content of the work 8

2. Pairwise Comparisons method 9

2.1. PC matrix 9

2.2. Weight vector 9

2.3. Inconsistency 10

3. Inconsistency indexes 11

3.1. Inconsistency indexes for complete matrices 11

3.1.1. Saaty index (*CI*) 11

3.1.2. Geometric consistency index (*GCI*) 11

3.1.3. Koczkodaj index (*K*) 12

3.1.4. Kazibudzki indexes (*MLTI*, *MLTI**, *CMLTI**) 12

3.1.5. Index of determinants (*PL*) 13

3.1.6. Kułakowski and Szybowski indexes ($I_1, I_2, I_\alpha, I_{\alpha,\beta}$) 13

3.1.7. Harmonic consistency index (*HCI*) 14

3.1.8. Golden and Wang index (*GW*) 14

3.1.9. Salo and Hämäläinen index (*CM*) 15

3.1.10. Cavallo and D'Apuzzo index (I_{CD}) 15

3.1.11. Relative error (*RE*) 15

3.2. Inconsistency indexes for incomplete matrices 16

4. Studies of the inconsistency indexes for incomplete matrices 19

4.1. Algorithm 19

4.1.1. Procedure steps: 19

4.1.2. Details of the algorithm 20

4.2. Implementation 20

4.2.1. Development environment 20
4.2.2. Implementation of the tests of the inconsistency indexes 20
4.2.3. Documentation 27
5. Results and discussion 29
5.1. Results 29
5.1.1. Tests taking into account different matrix sizes 29
5.1.2. Tests taking into account different degrees of incompleteness 30
5.1.3. Tests taking into account different levels of inconsistency 30
5.2. Discussion 30
5.2.1. Tests taking into account different matrix sizes 31
5.2.2. Tests taking into account different degrees of incompleteness 32
5.2.3. Tests taking into account different levels of inconsistency 33
5.2.4. General discussion 33
6. Summary 35
7. Appendix 37

1. Introduction

1.1. Pairwise Comparisons method

People have made decisions for ages. Some of them are very simple and come easily but others, more complicated, require deeper analysis. It happens when there are many compared objects, which are complex and the selection criterion is hard to measure precisely. Fortunately, the development of mathematics brought an interesting tool - *The Pairwise Comparisons (PC) Method*. The first case of using the method (in a very simple version) is the election system described by Ramond Llull [7] in the thirteenth century. Its rules were based on the fact that the candidates were pairwise compared with each other and the winner was the one who won in the largest number of direct comparisons. The method was reinvented in the eighteenth century by Condorcet and Bord [19] as they proposed it in their voting system. In the twentieth century, the method found the application in the theory of social choice, the main representatives of which were the Nobel prize winners Keneth Arrow [2] and Amartya Sen [26]. The current shape of the method was influenced by the changes introduced by Fechner [10] and then refined by Thrustone [30]. However, the breakthrough was the introduction to the method The Analytic Hierarchy Process (AHP) by Saaty [23], which allowed to compare many more complex objects and create a hierarchical structure.The primary aim of this paper is to check which method of calculating the inconsistency is the best in this case. In order to do it, a series of tests were carried out on various known inconsistency indexes, taking into account many different parameters: the matrix size, the amount of missing data and the level of inconsistency. The results of the research are included in this paper.

The PC method is based on the assumption that it is not worth comparing all objects at the same time. It is better to compare them in pairs and then gather the results together. Such pairwise comparisons are much more intuitive and natural for a human being. How can one be sure that these judgments are consistent? Or what to do if some comparisons are missing? In such a case, is it worth taking the PC method at all?

The answer to the first question is the concept of inconsistency introduced into the method. This paper tries to answer the next two questions - meaning to examine whether available methods for determining inconsistencies give reliable results when a part of the comparisons is missing.

1.2. The aim of the work

The main aim of this paper is to check which method of calculating the inconsistency is the best when some comparisons are missing. In order to do so, a series of tests was carried out on various known inconsistency indexes, taking into account many different parameters: the matrix size, the amount of missing data and the level of inconsistency. The tests were implemented in the *R* language which properly fit into numerical calculations.

It was not known from the beginning what the result of this work would be. It was considered that one or more existing inconsistency indexes would turn out to be appropriate for incomplete matrices also or that the tests would show that the inconsistency could not be calculated by these indexes. The results of the research are included in this study.

1.3. Content of the work

The work includes the theoretical part and the description of conducted experiments and their results. It consists of six chapters.

The second chapter shows the Pairwise Comparisons method and the problem of inconsistency. Understanding the basics is necessary to go into the further chapters.

In the third chapter, sixteen available methods for calculating the inconsistency for the PC matrices are presented.

The fourth chapter shows ideas which allow to perform tests. These are modifications of existing inconsistency indexes which enable to adjust them to incomplete matrices and the algorithm which tests the quality of the modified indexes.

The results of experiments are presented and discussed in the fifth chapter.

The last chapter contains summary, conclusions and ideas for future researches.

2. Pairwise Comparisons method

2.1. PC matrix

The *PC method* is used to choose the best alternative from a set of concepts. However, this goal is achieved by comparing in pairs. A numerical value is assigned to each pair. It not only determines which alternative is preferred but also informs about the strength of this preference. In this way, the finite set of concepts $C = \{c_1, \ldots, c_n\}$ is transformed into a *PC matrix* $M = (m_{ij})$, where $m_{i,j} \in \mathbb{R}$ and $i, j \in \{1, \ldots, n\}$. The PC matrix for n concepts is following:

$$M = \begin{pmatrix} 1 & m_{12} & \ldots & m_{1n} \\ m_{21} & 1 & \ldots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \ldots & 1 \end{pmatrix}$$

It is worth noting that the values m_{ij} and m_{ji} represent the same pair. Therefore, one should expect that $m_{ji} = \frac{1}{m_{ij}}$. If

$$\forall_{i,j \in \{1,\ldots,n\}}\, m_{ij} = \frac{1}{m_{ji}}, \tag{2.1}$$

the matrix is called a *reciprocal*.

Calculations on PC matrices are usually performed when the matrix contains all elements. For the matrix with dimension n by n there are $n(n-1)$ values. Such matrix is called *complete*. If one or more values are missing, the matrix is called *incomplete*.

2.2. Weight vector

The PC matrix provides data allowing the method for calculating the ranking vector . This vector defines the preferential values assigned by the function $\mu : C \to \mathbb{R}$ to each alternative c_i in C. The vector

$$\mu = [\mu(c_1), \ldots, \mu(c_n)]$$

formed in this way is called *the weight vector* or *the priority vector* (see Fig.2.1). By choosing the alternative with the highest preferential value in μ one can easily find which alternative win the ranking.

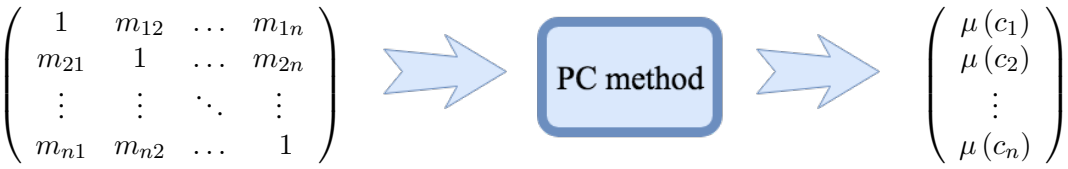


Fig. 2.1. Diagram of calculating *the weight vector* based on the article [18]

There are many ways to calculate the vector μ. Among the popular ones are the method using the matrix's eigenvalues or the method based on geometric means [24].

2.3. Inconsistency

The second important parameter describing the PC matrix is *consistency*. The matrix is consistent if

$$\forall_{i,j,k\in\{1,\ldots,n\}}\ m_{ik} = m_{ij}m_{jk}. \tag{2.2}$$

Three numbers that should meet this assumption are called a *triad*.

If the matrix is consistent and the weight vector μ is computed, then for each of the variables i, j, where $1 \leq i, j \leq n$ meets the equation:

$$m_{ij} = \frac{\mu_i}{\mu_j}. \tag{2.3}$$

In practice, it is very rare for a matrix M to be completely consistent. In the long history of the PC method, many methods were developed to calculate inconsistencies. Many of them are based directly on the definition of consistency (2.2), some methods use the eigenvalues of the matrix, others are based on the assumption that each fully consistent matrix fulfills the condition (2.2).

3. Inconsistency indexes

3.1. Inconsistency indexes for complete matrices

This subsection presents sixteen commonly used inconsistency indexes. In the next chapter their detailed description (including their formulas) will be modified, so that they can also work for incomplete matrices. Many of them were described and tested numerically in the article [4]. In all the cases, it is assumed that the PC matrix is reciprocal.

3.1.1. Saaty index (*CI*)

This is one of the most important and popular indexes. It was introduced by Saaty in his seminal work [22]. In order to determine inconsistency, the matrix's eigenvalues should be computed. The author used the dependence that the largest eigenvalue of the matrix is equal to its dimension if and only if the given matrix is completely consistent. On this assumption, he proposed the formula:

$$CI(M) = \frac{\lambda_{max} - n}{n-1}, \tag{3.1}$$

where λ_{max} is the principal eigenvalue of $n \times n$ matrix M.

3.1.2. Geometric consistency index (*GCI*)

Based on the principle (2.3) Craford and Williams defined geometric consistency index (*GCI*) [8]. The index was refined by Aguaròn and Moreno-Jimènez [1]. The authors assumed that in such a case a priority vector should be calculated using the geometric mean method. Consider (2.2) one can create a matrix:

$$E = \left[e_{ij} \middle| e_{ij} = m_{ij} \frac{w_j}{w_i} \right], \quad i, j = 1, ..., n. \tag{3.2}$$

The inconsistency index is calculated as follows:

$$GCI = \frac{2}{(n-1)(n-2)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} ln^2 e_{ij}. \tag{3.3}$$

3.1.3. Koczkodaj index (*K*)

One of the most popular inconsistency indexes was proposed by Koczkodaj [17]. It is based directly on the definition of consistency (2.2). The value of the inconsistency index for one triad of alternatives (c_i, c_j, c_k) was defined as:

$$K_{i,j,k} = \mathit{min} \left\{ \frac{1}{m_{ij}} \left| m_{ij} - \frac{m_{ik}}{m_{jk}} \right|, \frac{1}{m_{ij}} \left| m_{ik} - m_{ij} m_{jk} \right|, \frac{1}{m_{jk}} \left| m_{jk} - \frac{m_{ik}}{m_{ij}} \right| \right\}. \tag{3.4}$$

This formula was simplified by Duszak and Koczkodaj [9] and is given as:

$$K(\alpha, \beta, \gamma) = \mathit{min} \left\{ \left| 1 - \frac{\beta}{\alpha\gamma} \right|, \left| 1 - \frac{\alpha\gamma}{\beta} \right| \right\}, \ \ where\ \alpha = m_{ij},\ \beta = m_{ik},\ \gamma = m_{jk} \tag{3.5}$$

Then it was generalized [9] for $n > 2$. Finally, the inconsistency index has the following form:

$$K = \{ \mathit{max} \{ K(\alpha, \beta, \gamma) \mid 1 \leq i < j < k \leq n \}. \tag{3.6}$$

It is worth noting that not only does the coefficient find the most significant triad's inconsistency but also indicates the place in which it occurs.

3.1.4. Kazibudzki indexes (*MLTI*, *MLTI**, *CMLTI**)

Based on the Koczkodaj inconsistency index and observation that $ln\left(\frac{\alpha\gamma}{\beta}\right) = -ln\left(\frac{\beta}{\alpha\gamma}\right)$, Kazibudzki proposed several additional inconsistency indexes [16]. Instead of the formula for inconsistency of the triad [eq:k-abg], he introduced two new formulas:

$$\mathit{LTI}(\alpha, \beta, \gamma) = \left| ln\left(\frac{\alpha\gamma}{\beta}\right) \right|, \tag{3.7}$$

$$\mathit{LTI}^*(\alpha, \beta, \gamma) = ln^2\left(\frac{\alpha\gamma}{\beta}\right). \tag{3.8}$$

Based on the above equations, Kazibudzki proposed new indexes. The simplest ones use the geometric mean of the triads. Thus, new indexes could be written in the form:

$$\mathit{MLTI}(\mathit{LTI}) = \frac{1}{n} \sum_{i=1}^{n} \left[\mathit{LTI}_i(\alpha, \beta, \gamma) \right], \tag{3.9}$$

$$\mathit{MLTI}(\mathit{LTI}^*) = \frac{1}{n} \sum_{i=1}^{n} \left[\mathit{LTI}^*_i(\alpha, \beta, \gamma) \right]. \tag{3.10}$$

After further research Kazibudzki introduces another inconsistency index [15], again based on (3.8). It was defined as $CM(LTI^*) = \frac{MEAN[LTI^*(\alpha,\beta,\gamma)]}{1+MAX[LTI^*(\alpha,\beta,\gamma)]}$. Hence,

$$CM(LTI^*) = \frac{\frac{1}{n}\sum_{i=1}^{n}[LTI^*_i(\alpha,\beta,\gamma)]}{1 + max\left\{LTI^*_i(\alpha,\beta,\gamma)\right\}}. \tag{3.11}$$

3.1.5. Index of determinants (*PL*)

This index was proposed by Pelaez and Lamata [21] and is also based on the concept of triad. The authors noticed that PC matrices can be constructed on the basis of triads. Their determinant is closely related to the consistency of the matrix.

For every triad (m_{ik}, m_{ij}, m_{jk}) one can build a matrix in the form:

$$T_{ijk} = \begin{pmatrix} 1 & m_{ij} & m_{ik} \\ \frac{1}{m_{ij}} & 1 & m_{jk} \\ \frac{1}{m_{ik}} & \frac{1}{m_{jk}} & 1 \end{pmatrix}, \tag{3.12}$$

where $i < j < k$. The determinant of this matrix is:

$$det(M) = \frac{m_{ik}}{m_{ij}m_{jk}} + \frac{m_{ij}m_{jk}}{m_{ik}} - 2. \tag{3.13}$$

If the matrix is fully consistent, then $det(M) = 0$, else $det(M) > 0$. Based on the above considerations, the authors introduced the new inconsistency index:

$$CI^* = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{m_{ik}}{m_{ij}m_{jk}} + \frac{m_{ij}m_{jk}}{m_{ik}} - 2\right). \tag{3.14}$$

3.1.6. Kułakowski and Szybowski indexes ($I_1, I_2, I_\alpha, I_{\alpha,\beta}$)

Kułakowski and Szybowski proposed two further inconsistency indexes [20], which are also based on triads. They use the fact that the number of triads that can be found in a PC matrix is:

$$\binom{n}{3} = \frac{n!}{(n-3)!3!} = \frac{n(n-1)(n-2)}{6}. \tag{3.15}$$

The index is formulated as follows:

$$I_1 = \frac{6\sum_{t\in T} K(t)}{n(n-1)(n-2)}, \tag{3.16}$$

where $K(t)$ is the Koczkodaj index for triad $t = (\alpha, \beta, \gamma)$ of the set of all triads T.

The second inconsistency index is similar:

$$I_2 = \frac{6\sqrt{\sum_{t\in T} K^2(t)}}{n(n-1)(n-2)}. \tag{3.17}$$

Indexes can be combined with each other to create new coefficients. In this way Kułakowski and Szybowski proposed two new indexes. The first one is based on (3.4) and (3.16). This index allows to choose which value should have more impact on the result: the greatest inconsistency found or the average inconsistency of all triads. The new inconsistency index looks as follows:

$$I_\alpha = \alpha K + (1-\alpha) I_1, \tag{3.18}$$

where $0 \leq \alpha \leq 1$.

The second index expands the first one by (3.17):

$$I_{\alpha,\beta} = \alpha K + \beta I_1 + (1-\alpha-\beta) I_2. \tag{3.19}$$

3.1.7. Harmonic consistency index (*HCI*)

This index was introduced by Stein and Mizzi and it presents a completely new method of inconsistency computing [27]. At the beginning it requires the creation of an auxiliary vector $s = (s_1, ..., s_n)^T$, where n by n is the dimension of the matrix M, for which the index will be calculated. Each element of the vector s is the sum of values in one column of the matrix M. Hence,

$$s_j = \sum_{i=1}^{n} m_{ji} \quad \forall j. \tag{3.20}$$

The authors proved that if the matrix M is consistent, then $\sum_{j=1}^{n} s_j^{-1} = 1$. The formula for the mean harmonic looks as follows (Brunelli, 2015):

$$HM = \frac{n}{\sum_{j=1}^{n} \frac{1}{s_j}}. \tag{3.21}$$

The final formula for the inconsistency index was obtained by normalizing the above equation (3.21):

$$HCI = \frac{(HM(s) - n)(n+1)}{n(n-1)}. \tag{3.22}$$

3.1.8. Golden and Wang index (*GW*)

This index was introduced by Golden and Wang [11]. It assumes that the priority vector was calculated using the geometric mean method, then normalized to add up to 1. In this way vector $g^* = [g_1^*, ..., g_n^*]$ was obtained, where n by n is the dimension of the matrix M. The next step is to normalize each column of the matrix M. After this, the sum of the elements of each column in matrix

M is 1. The obtained matrix is marked with the symbol M^*. The inconsistency index is defined as follows:

$$GW = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\left|m_{ij}^* - g_i^*\right|. \tag{3.23}$$

3.1.9. Salo and Hämäläinen index (*CM*)

The index proposed by Salo and Hämäläinen [25] uses the definition of inconsistency (2.2). It requires, however, the creation of an auxiliary matrix, in which each element contains the smallest and largest discrepancy from consistency based on the formula (2.2). The index takes all triads into account:

$$R = (r_{ij})_{nxn} = \begin{pmatrix} [\underline{r}_{11}, \overline{r}_{11}] & \dots & [\underline{r}_{1n}, \overline{r}_{1n}] \\ \vdots & \ddots & \vdots \\ [\underline{r}_{n1}, \overline{r}_{n1}] & \dots & [\underline{r}_{nn}, \overline{r}_{nn}] \end{pmatrix}, \tag{3.24}$$

where $\underline{r_{ij}} = min\left\{m_{ik}m_{kj}\middle|k = 1, \dots, n\right\}$, $\overline{r_{ij}} = max\left\{m_{ik}m_{kj}\middle|k = 1, \dots, n\right\}$ and n by n is the dimension of the tested matrix M. A numerical example was presented in [3]. Based on the resulting matrix R, the authors proposed the following inconsistency index:

$$CM = \frac{2}{n(n-1)}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{\overline{r}_{ij} - \underline{r}_{ij}}{\left(1 + \overline{r}_{ij}\right)\left(1 + \underline{r}_{ij}\right)}. \tag{3.25}$$

3.1.10. Cavallo and D'Apuzzo index (I_{CD})

The authors Cavallo and D'Apuzzo based their index on triads but they conducted studies on a new path, generalizing them for linear, ordered abelian groups ([5], [6]). Thanks to this, the index can be used also with other relations [4]. Index for relation *max* can be presented in the form of a formula:

$$I_{CD} = \prod_{i=1}^{n-2}\prod_{j=i+1}^{n-2}\prod_{k=j+1}^{n}\left(max\left\{\frac{m_{ik}}{m_{ij}m_{jk}}, \frac{m_{ij}m_{jk}}{m_{ik}}\right\}\right)^{\frac{1}{\binom{n}{3}}}. \tag{3.26}$$

3.1.11. Relative error (*RE*)

This index, proposed by Barzaili [14], requires calculation of the weight vector using the arithmetic mean method for each row and creation of two additional matrices. Thus, the weight vector is

$$w_i = \frac{1}{n}\sum_{j=1}^{n} m_{ij},$$

where n by n is the dimension of the matrix. The two auxiliary matrices are calculated according to the formulas:

$$C = (c_{ij}) = (w_i - w_j)$$

$$E = (e_{ij}) = (m_{ij} - c_{ij})$$

Ultimately, the formula for the Relative error is the following:

$$RE(M) = \frac{\sum_{ij} e_{ij}^2}{\sum_{ij} m_{ij}^2}. \tag{3.27}$$

3.2. Inconsistency indexes for incomplete matrices

There are no inconsistency indexes for incomplete matrices. However, Those presented in chapter 3 could be used in such cases. It requires usually a slight modification of the index definition or calculation only for selected data. The ways in which the examined indexes were adjusted to be able to deal with incomplete matrices are presented below.

Saaty index:

The input matrix is modified using the method proposed by Harker [12]. It means that values $c+1$, where c is the number of non-empty elements in a given row, are placed on the diagonal.

Geometric consistency index:

During calculating the weight vector by the geometric mean, empty values are omitted. Additionally, in the formula (3.3) only non-empty elements e_{ij} are used. The reason for this exclusion is that the domain of the logarithmic function is $\mathbb{R}^+$.

Koczkodaj index, Kazibudzki indexes, Index of determinants:

Only those triads which do not contain empty values are taken into account.

Kułakowski and Szybowski indexes:

Only those triads which do not contain empty values are taken into account. In addition, the number of triads is no longer calculated according to the formula (3.15) but determined directly by counting the number of triads.

Harmonic consistency index:

No modification.

Golden and Wang index:

During calculating the weight vector by the geometric mean empty values are omitted.

Salo and Hämäläinen:

No modification.

Cavallo and D'Appuzo:

During calculating the product (3.26) empty values are omitted.

Relative index:

No modification.

Proposed methods of adjustments of indexes allow to apply them to incomplete matrices. However, they do not guarantee the best results. It means that further experiments are required. The existing indexes can be modified in many different ways. In this paper emphasis is put on the fact that the adjustments were slight. The aim of this study is to examine existing indexes, not to create new ones.

4. Studies of the inconsistency indexes for incomplete matrices

The presented inconsistency indexes were tested utilizing the Monte Carlo method. Their goal was to select those indexes which would give reliable results for incomplete matrices. Therefore, it was decided that the measure of the indexes' quality would be the *relative error* (expressed as percentage), which took into account the value of the index for a full, inconsistent matrix and the value of the index for the same matrix after partial decomposition. To be sure that the results were fair, all indexes were tested on the same set of matrices. The different sizes of the matrices, the levels of incompleteness and the levels of inconsistency were taken into account. Then, in order to compare the indexes easily and to select the best ones, the results were averaged using the arithmetic mean. The simulation algorithms *SA|1|* and *SA|2|* from the article [15] was used while building the algorithm to solve the problem.

4.1. Algorithm

4.1.1. Procedure steps:

1. Randomly generate a vector $w = [w_1, ..., w_n]$ and a consistent PC matrix associated with it $PCM = (m_{ij})$, where $m_{ij} = \frac{w_i}{w_j}$.
2. Disrupt the matrix by multiplying its elements (excluding the diagonal) by the value of d, randomly selected from the range $\left(\frac{1}{x}, x\right)$.
3. Replace the values m_{ij}, where $i < j$ by the values m_{ji}.
4. Calculate the values of index with all the methods for the created matrix.
5. Remove some of the values from the matrix. The level of incompleteness should be $g\%$.
6. Calculate the values of inconsistencies by all methods for the incomplete matrix.
7. Calculate the relative error for each index.
8. Repeat steps 1 to 7 X_1 times.
9. Calculate the average relative error for each inconsistency index for the PC matrix.
10. Repeat steps 1 to 9 X_2 times.

11. Calculate the average relative error for each index by averaging the values obtained in step 9.

4.1.2. Details of the algorithm

The above algorithm was used to carry out for values $X_1 = 100$, $X_2 = 100$. Tests were started for values d in range $(1.1, 1.2, ..., 4)$ and then the results were averaged. It means that the average relative error of one index was calculated on the basis of 4000 matrices, each of which decomposed randomly 100 times. It gave together 400000 tests on how good the index was.
In addition, tests were carried out for various sizes of matrices.

The results are divided into two parts:

1. A constant degree of incompleteness, different size of the matrix.
2. Different degrees of incompleteness, constant size of the matrix.

The aim of a such division is to pay attention to how the inconsistency indexes behave when the size of the matrix and the degree of incompleteness are changing. The results of the research are presented below.

4.2. Implementation

4.2.1. Development environment

Tests of indexes were developed in R language which is appropriate for numerical calculations (see [29]). It contains sixteen of functions which support operations on matrices and vectors. Integrated development environment (IDE), named *RStudio*, (see [28]) was used during the implementation. This tool allows to create own packages which contain not only code but also documentation and information about licence and author. Using *RStudio* package `indexesForIncomplete` was created. The most important part of this package is the file `indexes.R` which performs calculations necessary to test the indexes. *RStudio* supports programmer's work by syntax highlighting, built-in console, easy documentation searching and many others. The program is available on common operating systems. Before using *RStudio* one has to install *R* programming language (see [13]).

4.2.2. Implementation of the tests of the inconsistency indexes

The implementation consists of two steps:

1. Implementation of functions which calculate inconsistency indexes for a given matrix (full or incomplete).
2. Implementation of functions which study indexes for different matrices and collect all the results of these tests.

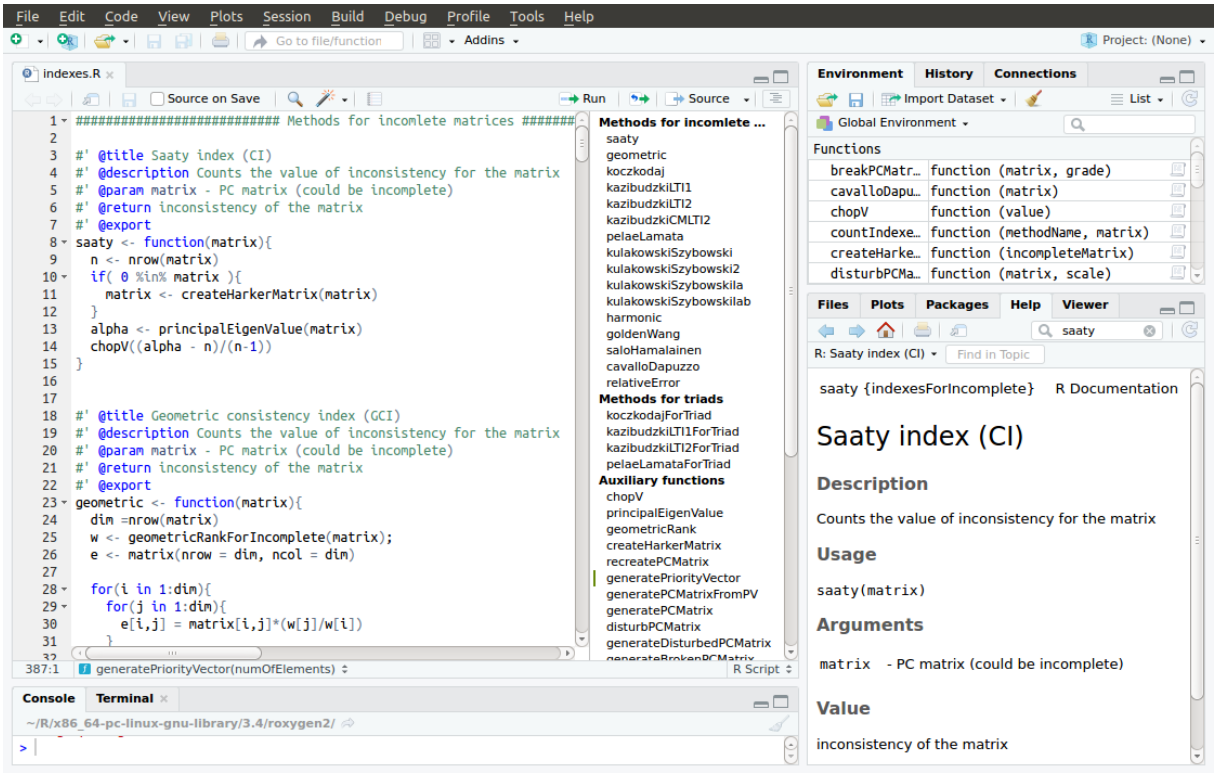


Fig. 4.1. Program *RStudio*

4.2.2.1. Implementation of the inconsistency indexes

Sixteen functions which calculate the inconsistency indexes using methods are described in chapter 3. The functions were written in such a way that allows handling both full and incomplete matrices. The wrong matrices (it means nonreciprocal or inconsistent PC matrices) were not taken into account. Each of these function has only one parameter - *PC matrix*. Exceptions are the two methods implementing Kulakowski and Szybowski index which additionally takes parameters α and β. The result of each function is a value of the inconsistency index. The functions were extended by comments which inform about the name of the index related to given function, parameters and returned value. It allows to read and modify the code easily. Several examples of the functions are presented below.

It is worth drawing attention to a function which is called within the functions intended for indexes based on triads. This function generates triads from a matrix and next returns inconsistency for each of them. The way to calculate inconsistency for one triad depends on a function passed as a parameter.

```r
#' @title Saaty index (CI)
#' @description Counts the value of inconsistency for the matrix
#' @param matrix - PC matrix (could be incomplete)
#' @return inconsistency of the matrix
#' @export
saaty <- function(matrix){
  n <- nrow(matrix)
  if( 0 %in% matrix ){
    matrix <- createHarkerMatrix(matrix)
  }
  alpha <- principalEigenValue(matrix)
  chopV((alpha - n)/(n-1))
}
```

Fig. 4.2. The implementation of Saaty index

```r
#' @title Koczkodaj matrix inconsistency
#' @description Counts the value of inconsistency for the matrix
#' @param matrix - PC matrix (could be incomplete)
#' @return inconsistency of the matrix
#' @export
koczkodaj <- function(matrix){
  triadsAndIdxs <- countIndexesForTriads("koczkodajForTriad", matrix)
  chopV(max(triadsAndIdxs))
}
```

Fig. 4.3. The implementation of Koczkodaj index

```r
#' @title Kulakowski and Szybowski consistency index (Ia)
#' @description Counts the value of inconsistency for the matrix
#' @param matrix - PC matrix (could be incomplete)
#' @return inconsistency of the matrix
#' @export
kulakowskiSzybowskiIa <- function(matrix, alfa, beta=0){
  triadsAndIdxs <- countIndexesForTriads("koczkodajForTriad", matrix)
  chopV(alfa*max(triadsAndIdxs) + (1-alfa)*mean(triadsAndIdxs))
}
```

Fig. 4.4. The implementation of Kulakowski and Szybowski index (I_α)

```r
#' @title Salo and Hamalainen consistency index (SH)
#' @description Counts the value of inconsistency for the matrix
#' @param matrix - PC matrix (could be incomplete)
#' @return inconsistency of the matrix
#' @export
saloHamalainen <- function(matrix){
  n <- nrow(matrix)
  sum <- 0

  rMin <- matrix(nrow = n, ncol = n, data = 0)
  rMax <- matrix(nrow = n, ncol = n, data = 0)

  for(i in 1:n){
    for(j in 1:n){
      c <- rep(0,n)
      for(k in 1:n){
        c[k] = matrix[i,k]*matrix[k,j]
      }
      rMin[i,j] = min(c)
      rMax[i,j] = max(c)
    }
  }

  for(i in 1:(n-1)){
    for(j in (i+1):n){
      sum <- sum + (rMax[i,j]-rMin[i,j])/((1+rMax[i,j])*(1+rMin[i,j]))
    }
  }

  cm <- 2/(n*(n-1))*sum
  chopV(cm)
}
```

Fig. 4.5. The implementation of Salo and Hämäläinen index

```r
#' @title Generates inconsistency indexes for each triad from the matrix
#' @param methodName - name of the method which computes inconsistency index for triad
#' @param matrix - PC matrix
#' @return vector of inconsistency indexes for triads
countIndexesForTriads <- function(methodName, matrix){
  triads <- generateTriads(matrix)
  if(is.null(dim(triads)) && length(triads) == 3){
    triads <- matrix(triads, 3, 1)
  }
  triadIdxs <- apply(triads, 2, methodName)
  triadIdxs
}
```

Fig. 4.6. The implementation of the function `countIndexesForTriads` which calculates inconsistency for each triad of a specified matrix

4.2.2.2. Implementation of tests

In the second step, functions calculating the quality of the indexes for incomplete matrices, were created. Functions, which generate specified matrices, play an important role. PC matrices are created depending on the size, the level of inconsistency and the degree of incompleteness.

```
#' @title Generates PC Matrix on the basis of the size
#' @param numOfElements - number of elements of the matrix
#' @return reciprocal PC matrix
generatePCMatrix<- function(numOfElements){
  priorityVector <- generatePriorityVector(numOfElements)
  generatePCMatrixFromPV(priorityVector)
}
```

Fig. 4.7. The implementation of the function `generatePCMatrix` which generates the PC matrix depending on matrix size

```
#' @title Breaks PC Matrix
#' @description Breaks PC Matrix to obtain reciprocal, incomplete matrix
#' @param  matrix - PC matrix
#' @param grade - percentage of the value to be removed (not applicable to the diagonal)
#' @return incomplete PC matrix
breakPCMatrix<- function(matrix, grade){

  dim <- nrow(matrix)
  numOfEmptyElements <- grade*0.01*(dim*(dim-1))

  while(numOfEmptyElements > 1){
    rowToClear <- sample(1:dim, 1)
    colToClear <- sample(c(1:dim)[-rowToClear], 1)

    if(matrix[rowToClear, colToClear] != 0) {
      matrix[rowToClear, colToClear] <- matrix[colToClear, rowToClear] <- 0
      numOfEmptyElements <- numOfEmptyElements - 2
    }
  }

  matrix

}
```

Fig. 4.8. The implementation of the function `breakPCMatrix` which breaks up the PC matrix regarding incompleteness depending on given degree of incompleteness

The last part of the functions relates to testing how big relative error occurs for inconsistency indexes after deleting some of the values. To begin with functions which test one index. They consider matrix size, the level of inconsistency, the degree of incompleteness and the number of attempts which are performed for a given matrix.

Then functions, which perform tests for each index, were implemented based on the same set of matrices. Thus, the results are reliable and each index is considered the same way.

```
#' @title Disturbs the PC matrix
#' @description Disturbs the PC matrix in order to obtain inconsistency
#' @param matrix - PC matrix
#' @param scale - extend of disorders. This parametr is the upper limit
#' of the interval that is used to scale the elements. The lower limit
#' is defined as 1 / scale
#' @return disturbed PC matrix
disturbPCMatrix<- function(matrix, scale){
  dim = nrow(matrix)

  for(r in 1:dim-1)
    for(c in (r+1):dim)
      matrix[r,c] <- runif(1, min = 1/scale, max = scale) * matrix[r,c]

    diag(matrix) <- 1
    recreatePCMatrix(matrix)
}
```

Fig. 4.9. The implementation of the function `disturbPCMatrix` which disturbs the PC matrix regarding inconsistency depending on given level of inconsistency

```
#' @title Explore the incomplete PC matrixes for every method and scale <1.1, 1.2, ... , 4.0>
#' @description Examines what is the relative error between the full matrixes and indomplete matrixes.
#' @param numOfElements - dimension of tested matrix
#' @param gradeOfIncomplete -  percentage of the value to be removed (not applicable to the diagonal)
#' @param numOfAttempts - number of tested matrixes
#' @param numOfAttemptsForOneMatrix - number of test cases for each matrix
#' @param alfa - a parameter for kulakowskiSzybowskiIa method
#' @param beta - a parameter for kulakowskiSzybowskiIab method
#' @return average value of the relative error between the full matrix and indomplete matrix
test <- function(numOfElements, gradeOfIncomplete, numOfAttempts, numOfAttemptsForOneMatrix, alfa=0, beta=0){

  results <- matrix(nrow=31, ncol=16, data=0)
  counter <- 1

  for(i in seq(1.1, 4, 0.1)){
    print(counter)
    results[counter,] <- monteCarloOnTheSameMatrix(numOfElements, i, gradeOfIncomplete, numOfAttempts,
                                                   numOfAttemptsForOneMatrix, alfa, beta)
    counter <- counter+1
  }

  for(i in 1:16){
    results[31, i] = sum(results[,i])/30
  }

  results
}
```

Fig. 4.10. The implementation of the function `test` which tests all indexes regarding given matrix size and the degree of incompleteness

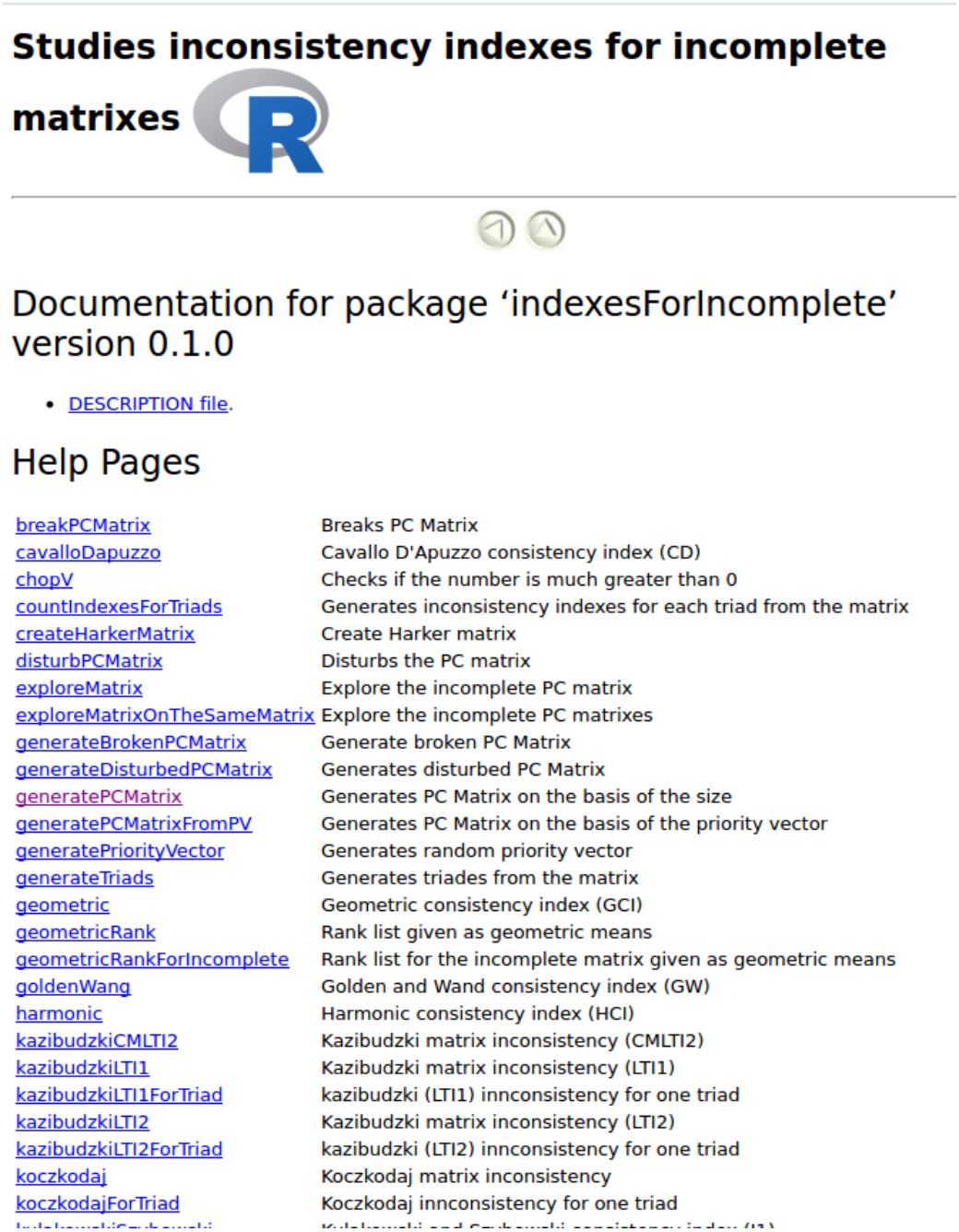
```
#' @title Explore the incomplete PC matrix
#' @description Examines what is the relative error between the full matrix and indomplete matrix
#' @param methodName - a name of the method which is tested
#' @param scale - extend of disorders. This parametr is the upper limit of the interval that is used
#' to scale the elements. The lower limit is defined as 1 / scale
#' @param numOfElements - dimension of tested matrix
#' @param gradeOfIncomplete -  percentage of the value to be removed (not applicable to the diagonal)
#' @param numOfAttempts - number of test cases
#' @param alfa - a parameter for kulakowskiSzybowskiIa and kulakowskiSzybowskiIab method
#' @param beta - a parameter for kulakowskiSzybowskiIab method
#' @return average value of the relative error between the full matrix and indomplete matrix
exploreMatrix <- function(methodName, scale, numOfElements, gradeOfIncomplete, numOfAttempts, alfa=0, beta=0) {

  matrix <- generateDisturbedPCMatrix(numOfElements, scale)
  n <- ncol(matrix)
  if(alfa==0 && beta==0){
    realIdx <- methodName(matrix)
  } else {
    realIdx <- methodName(matrix, alfa, beta)
  }

  vectorOfIdsx <- integer(numOfAttempts)

  for( i in 1:numOfAttempts ) {
    brokenMatrix <- matrix(nrow = n, ncol = n, data = 0)
    numOfTriads <- length(generateTriads(brokenMatrix))/3

    while( (0 %in% (matrix %^% (n-1))) || numOfTriads==0){
      brokenMatrix <- breakPCMatrix(matrix, gradeOfIncomplete)
      numOfTriads <- length(generateTriads(brokenMatrix))/3
    }

    if(alfa==0 && beta==0){
      idx <- methodName(brokenMatrix)
    } else {
      idx <- methodName(brokenMatrix, alfa, beta)
    }

    vectorOfIdsx[i] <- abs(realIdx - idx)
  }

  incompleteIdx <- mean(vectorOfIdsx)

  abs(incompleteIdx/realIdx*100)
}
```

Fig. 4.11. The implementation of the function `exploreMatrix` which tests given inconsistency index

4.2.3. Documentation

Software written for the purpose of testing of the inconsistency indexes for incomplete matrixes contains comments. They were used to generate the documentation. The `roxygen2` package was used for this purpose.

Exemplary portions of the documentation are presented below.



Studies inconsistency indexes for incomplete matrixes

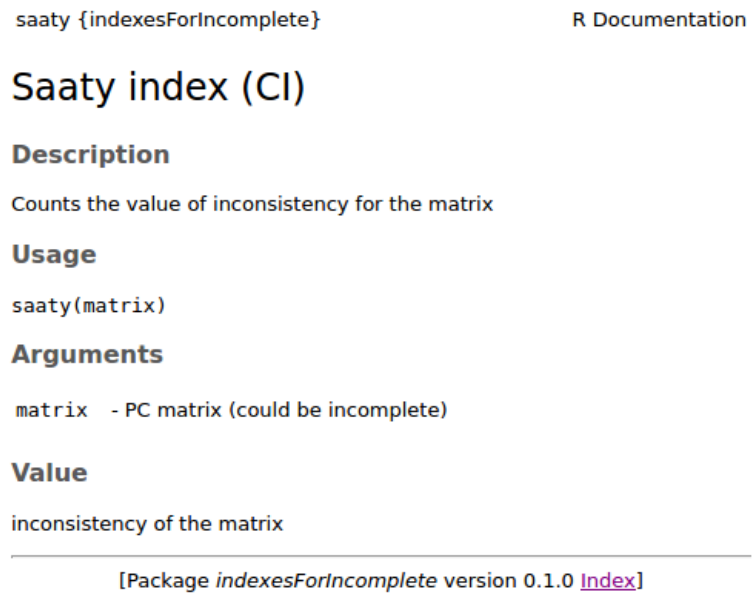
Documentation for package ‘indexesForIncomplete’ version 0.1.0

- DESCRIPTION file.

Help Pages

breakPCMatrix	Breaks PC Matrix
cavalloDapuzzo	Cavallo D'Apuzzo consistency index (CD)
chopV	Checks if the number is much greater than 0
countIndexesForTriads	Generates inconsistency indexes for each triad from the matrix
createHarkerMatrix	Create Harker matrix
disturbPCMatrix	Disturbs the PC matrix
exploreMatrix	Explore the incomplete PC matrix
exploreMatrixOnTheSameMatrix	Explore the incomplete PC matrixes
generateBrokenPCMatrix	Generate broken PC Matrix
generateDisturbedPCMatrix	Generates disturbed PC Matrix
generatePCMatrix	Generates PC Matrix on the basis of the size
generatePCMatrixFromPV	Generates PC Matrix on the basis of the priority vector
generatePriorityVector	Generates random priority vector
generateTriads	Generates triades from the matrix
geometric	Geometric consistency index (GCI)
geometricRank	Rank list given as geometric means
geometricRankForIncomplete	Rank list for the incomplete matrix given as geometric means
goldenWang	Golden and Wand consistency index (GW)
harmonic	Harmonic consistency index (HCI)
kazibudzkiCMLTI2	Kazibudzki matrix inconsistency (CMLTI2)
kazibudzkiLTI1	Kazibudzki matrix inconsistency (LTI1)
kazibudzkiLTI1ForTriad	kazibudzki (LTI1) innconsistency for one triad
kazibudzkiLTI2	Kazibudzki matrix inconsistency (LTI2)
kazibudzkiLTI2ForTriad	kazibudzki (LTI2) innconsistency for one triad
koczkodaj	Koczkodaj matrix inconsistency
koczkodajForTriad	Koczkodaj innconsistency for one triad

Fig. 4.12. The portion of the documentation: general view



saaty {indexesForIncomplete} R Documentation

Saaty index (CI)

Description

Counts the value of inconsistency for the matrix

Usage

```
saaty(matrix)
```

Arguments

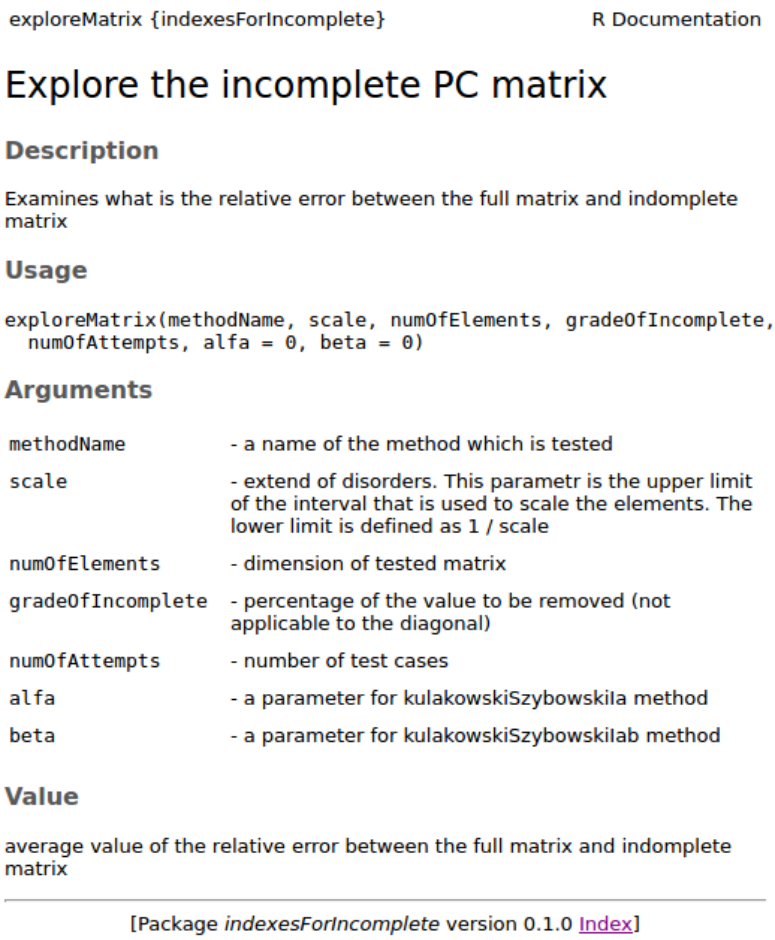
matrix	- PC matrix (could be incomplete)

Value

inconsistency of the matrix

[Package *indexesForIncomplete* version 0.1.0 Index]

Fig. 4.13. The portion of the documentation: function `saaty`



exploreMatrix {indexesForIncomplete} R Documentation

Explore the incomplete PC matrix

Description

Examines what is the relative error between the full matrix and indomplete matrix

Usage

```
exploreMatrix(methodName, scale, numOfElements, gradeOfIncomplete,
  numOfAttempts, alfa = 0, beta = 0)
```

Arguments

methodName	- a name of the method which is tested
scale	- extend of disorders. This parametr is the upper limit of the interval that is used to scale the elements. The lower limit is defined as 1 / scale
numOfElements	- dimension of tested matrix
gradeOfIncomplete	- percentage of the value to be removed (not applicable to the diagonal)
numOfAttempts	- number of test cases
alfa	- a parameter for kulakowskiSzybowskiIa method
beta	- a parameter for kulakowskiSzybowskiIab method

Value

average value of the relative error between the full matrix and indomplete matrix

[Package *indexesForIncomplete* version 0.1.0 Index]

Fig. 4.14. The portion of the documentation: function `exploreMatrix`

5. Results and discussion

5.1. Results

The tests were performed using Operating System *Ubuntu 16.04 LTS* and IDE *RStudio*. The partial results were saved in a spreadsheet and analyzed. It helped to draw the right conclusions. The results are presented in the following section.

5.1.1. Tests taking into account different matrix sizes

The results of the first part of the tests are presented in this subsection.

Table 5.1. Relative error of the inconsistency indexes for incomplete matrices with constant degree of incompleteness $g = 15\%$ and variable matrix size given as n.

Index	$n = 4$	$n = 6$	$n = 8$	$n = 10$	$n = 15$	Mean	Rank
CI	33.41	19.82	18.78	19.16	17.37	21.71	10
GCI	616.68	124.73	77.94	68.62	39.13	185.42	13
K	**13.86**	**3.69**	**2.14**	**1.62**	**0.80**	**4.42**	**1**
MLTI	24.80	10.21	6.62	4.97	2.73	9.87	6
*MTLI**	42.31	17.93	11.88	9.03	5.03	17.24	8
*CMLTI**	35.40	17.07	13.26	11.20	6.81	16.75	7
PL	44.65	19.90	13.46	10.36	5.84	18.84	9
I_1	20.34	7.68	4.88	3.63	1.96	7.70	5
I_2	44.61	26.05	27.12	29.64	28.46	31.18	11
I_α	16.47	5.18	3.09	2.27	1.16	5.63	3
$I_{\alpha,\beta}$	17.40	4.89	2.81	2.04	1.01	5.63	2
HCI	9 573.02	1 577.49	1 127.33	1 066.35	866.00	2 842.04	15
GW	115.92	54.37	43.90	43.16	36.26	58.72	12
CM	381.57	205.06	176.11	160.06	136.55	211.87	14
I_{CD}	16.94	6.85	4.46	3.36	1.87	6.70	4
RE	1 792.64	226 313.60	746.21	100.87	20.42	45 794.75	16

5.1.2. Tests taking into account different degrees of incompleteness

The results of the second part of the tests are presented in this subsection.

Table 5.2. Relative error of the inconsistency indexes for incomplete matrices with various degrees of incompleteness and constant matrix size $n = 8$.

Index	$g = 4\%$	$g = 7\%$	$g = 14\%$	$g = 25\%$	$g = 50\%$	Mean	Rank
CI	4.71	9.40	18.78	32.89	65.56	26.27	10
GCI	23.60	48.44	86.61	135.68	207.99	100.46	13
K	**0.48**	**0.99**	**2.17**	**4.52**	16.41	4.92	2
MLTI	2.90	4.31	6.64	10.05	23.09	9.40	6
*MLTI**	5.12	7.71	11.91	18.08	40.77	16.72	7
*CMLTI**	5.16	8.05	13.34	22.03	61.16	21.95	9
PL	5.73	8.72	13.52	20.54	45.64	18.83	8
I_1	2.17	3.18	4.91	7.43	17.30	7.00	5
I_2	5.90	12.22	27.12	56.71	202.27	60.84	12
I_α	1.20	1.88	3.12	5.13	13.97	5.06	3
$I_{\alpha,\beta}$	1.00	1.65	2.84	4.74	**12.97**	**4.64**	**1**
HCI	291.74	544.60	1 152.26	1 962.25	3 995.58	1 589.29	16
GW	14.23	25.23	46.18	68.83	98.24	50.54	11
CM	88.40	137.70	180.17	182.54	148.74	147.51	14
I_{CD}	1.95	2.91	4.46	6.81	16.11	6.45	4
RE	18.99	20.81	206.74	68.98	1 056.96	274.50	15

5.1.3. Tests taking into account different levels of inconsistency

The inconsistency indexes tests which gave the best results are presented below. The detailed results are placed in the Appendix.

5.2. Discussion

The task of this work was not only to analyze the inconsistency indexes for incomplete matrixes but also to try to assess these indexes in this respect. Conclusions were divided into sections corresponding to the tests carried out and described in the previous section. Analyzing the above tables, one can draw a conclusion that the matrix size and the level of incompleteness have a significant impact on the results. The detailed results (included in Appendix) allow to conclude that the degree of incompleteness is also crucial.

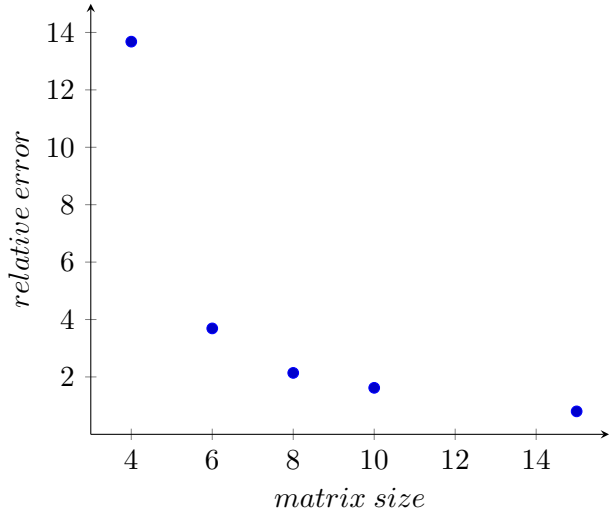
Table 5.3. Relative error of the inconsistency indexes for incomplete matrices with various degrees of inconsistency (given as d), constant matrix size $n = 8$ and constant level of incompleteness $g = 15\%$.

d	K	$MLTI$	I_1	I_α	$I_{\alpha,\beta}$	I_{CD}
1.1	4.09	6.38	6.13	4.52	4.35	**0.49**
1.2	3.68	6.45	5.98	4.26	4.06	**0.96**
1.3	3.58	6.55	5.87	4.16	3.96	**1.37**
1.4	3.55	6.62	5.77	4.12	3.91	**1.80**
1.5	3.19	6.35	5.42	3.80	3.59	**2.02**
1.6	3.26	6.57	5.46	3.84	3.64	**2.39**
1.7	3.03	6.59	5.35	3.71	3.48	**2.81**
1.8	**2.47**	6.45	5.13	3.31	3.04	3.07
1.9	**2.47**	6.65	5.26	3.35	3.06	3.34
2.0	**2.20**	6.66	5.13	3.16	2.85	3.64
2.1	**2.38**	6.51	4.93	3.21	2.93	3.83
2.2	**2.16**	6.52	4.81	3.04	2.76	4.09
2.3	**2.14**	6.66	4.89	3.07	2.80	4.25
2.4	**2.00**	6.67	4.85	2.98	2.67	4.49
2.5	**1.96**	6.59	4.73	2.91	2.63	4.71
2.6	**1.73**	6.62	4.72	2.81	2.48	4.87
2.7	**1.94**	6.79	4.73	2.96	2.67	5.14
2.8	**1.75**	6.67	4.64	2.78	2.47	5.24
2.9	**1.56**	6.67	4.54	2.68	2.38	5.60
3.0	**1.82**	6.71	4.68	2.84	2.51	5.56
3.1	**1.47**	6.69	4.48	2.62	2.30	5.74
3.2	**1.46**	6.52	4.40	2.59	2.28	5.75
3.3	**1.48**	6.85	4.57	2.65	2.35	6.14
3.4	**1.38**	6.60	4.30	2.53	2.23	6.15
3.5	**1.17**	6.60	4.25	2.40	2.10	6.50
3.6	**1.25**	6.73	4.39	2.50	2.18	6.65
3.7	**1.18**	6.83	4.41	2.48	2.19	6.77
3.8	**1.24**	6.59	4.17	2.40	2.09	6.68
3.9	**1.28**	6.73	4.25	2.47	2.19	6.79
4.0	**1.11**	6.52	4.17	2.35	2.06	6.79
Mean	**2.13**	6.61	4.88	3.08	2.81	4.45

5.2.1. Tests taking into account different matrix sizes

Along with growth of the matrix size, examined relative error decreases. Considering the best indexes, the relative error is between 13 and 17 percent for small matrices (size 4) so it is relatively big. However, it decreases rapidly when the size of PC matrix increases (see Fig. 5.1). For the matrix of size 6 there are indexes which give the results in range $3 - 5\%$, what is an acceptable score. For PC matrices with a significant size (over 10 concepts), the relative error is just about $1 - 2\%$, what is a really good score.

Fig. 5.1. Relative error of the index K for incomplete matrices with constant degree of incompleteness g = 15%



Definitely, the best index found out is out the index proposed by Koczkodaj (K). It wins in each test regardless of the matrix size. It gives really satisfying results for matrices of size 6 and more. Good results are obtained also through the indexes $I_{\alpha,\beta}$, I_α and I_{CD}. It seems that these indexes are stable with respect to incompleteness. Based on the evidence one should reject the indexes *RE*, *HCI*, *CM*, *GCI* and *GW* for incomplete matrices. Results obtained through these methods show that the score of examined inconsistency for incomplete matrices can differ significantly from the right values.

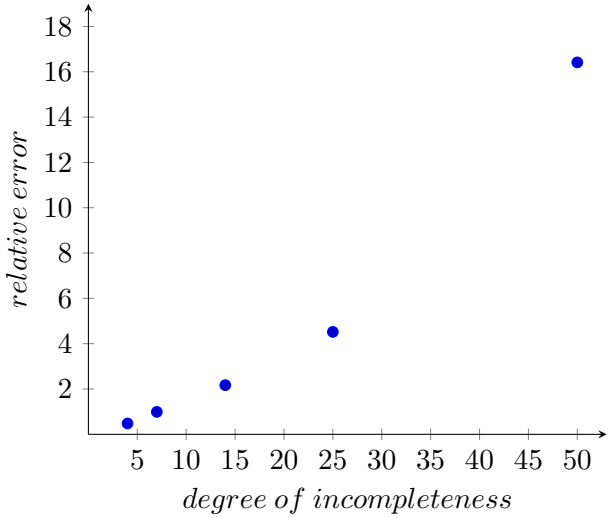
It is worth noticing that the presented results involve matrices with the degree of incompleteness equal to 15%. It means that 7 comparisons are missing for matrices with size $n = 10$. The results will be different from presented in Table 5.1 for other degrees of incompleteness. The impact of the degree of incompleteness on tested relative error is shown in consecutive experiments.

5.2.2. Tests taking into account different degrees of incompleteness

Definitely, the indexes *K* and $I_{\alpha,\beta}$ came out the best indexes. The first one wins in most cases and gives really satisfying results. The second one improves along with the growth of the degree of incompleteness. It obtained a significant advantage over other methods in the last test, what caused that the average value turned out to be the lowest. The good results were obtained also through the indexes I_α and I_{CD}. It seems that these indexes are stable with respect to incompleteness. Absolutely one should reject the indexes *HCI*, *RE*, *CM*, *GCI*. Results obtained through these methods show that the score of examined inconsistency for incomplete matrices can differ significantly from the right values.

It is worth noticing that the presented results involve matrices with the size 8%. The results will be different from presented in Table 5.2 for other matrix sizes. The impact of the degree of incompleteness on tested relative error is shown in the previous experiments.

Fig. 5.2. Relative error of the index K for incomplete matrices with constant matrix size n = 8



5.2.3. Tests taking into account different levels of inconsistency

All tests for different matrix sizes and degrees of incompleteness were performed taking into account various levels of inconsistency. The scale of this incompleteness is proportional to the value d included in Table 5.3 This table shows detailed results for which the average values are presented in Table 5.1 It is worth taking a closer look at these scores. They point out that the quality of the inconsistency indexes is determined also by the level of inconsistency.

The lowest values of the relative error was obtained by K, what is discussed before. It should be noticed and emphasized that it gives the best results only from particular moment in which the level of inconsistency in matrix begins to grow. For small inconsistencies, the most satisfying index is I_{CD}. This is an interesting regularity which indicates that the choice of an inconsistency index for incomplete matrices should depend on the level of inconsistency.

It is worth noticing that this regularity repeats in each of the performed tests. For small level of inconsistency, regardless of the matrix size and the degree od incompleteness, I_{CD} comes out to be the best. The results presented in the Appendix confirm it.

5.2.4. General discussion

Certainly, the tests managed to show that the error increases with a growth of the level of incompleteness. At the same time, it decreases when the size of the matrix increases. However, the most important question was about which indexes cope well with incomplete matrices. The Koczkodaj index wins in 9 out of 10 tests and its average error in both cases turns out to be the lowest (below 5%). The next places are occupied by two indexes introduced by Kułakowski and Szybowski and Cavallo and D'Apuzzo index. It is worth noting that all of these indexes are based on triads.

A question about what makes the Koczkodaj index giving such good results and whether it is worth using, may arise. One should return to the definition of this index (3.4) and notice that it is equal to the value of the most inconsistency triad. Therefore, if the level of incompleteness is low, there is a good chance that after deletion of some values from the matrix and recalculation at the index the value of it will not change at all. It will only change if the element included in the most inconsistent triad is removed. However, in many cases, the examination of the full matrix and the matrix after removing some values give exactly the same results (the error is 0%). This is the only index of this kind among those presented in the paper.

If one uses the Koczkodaj index, one may be worried that the removed comparison belonged to the most inconsistent triad. In such a case, it is difficult to predict what error will be contained in the index of the incomplete matrix. It seems that one should pay particular attention to the indexes proposed by Kułakowski and Szybowski. First of them (I_1) averages the inconsistencies of the triads. Therefore, it is safe and gives good results (in both tests it took the fifth place and achieved an error below 8%). From this perspective, another index suggested by the same authors (I_α) turns out to be very interesting. In the tests it took the third place. It has the parameter α allowing to determine the effect of the greatest inconsistency of the triad (α) and the average $(1-\alpha)$. In the tests carried out, the parameter α was 0.4.

6. Summary

The goal of the work - testing common inconsistency indexes for incomplete PC matrices was achieved. Sixteen different methods for calculating inconsistency were examined. The performed tests took into account many factors which could have an impact on results. It considered matrices with different sizes and the levels of inconsistency. They were checked for various degree of incompleteness (from 4% to 50%). All results are gathered and presented in the tables.

All tests were implemented in the R language which is appropriate for numerical calculations and it fulfilled the task. Functions, which are responsible for calculations, are described and documented in details. They can compute the inconsistency indexes for both full and incomplete matrices. If applicable, one can implement another indexes easily.

The tests indicate that some of existing indexes are up to calculating inconsistency for incomplete matrices after slight modifications. The methods of these modifications are presented in this work. Obviously there are many different possibilities to make changes in these indexes. Perhaps they can give even more favorable results. However, the purpose of this work was testing only existing indexes, therefore, it was decided not to make many modifications. The most reliable results were achieved by K, I_α, $I_{\alpha,\beta}$ and I_{CD}. Selection one of these indexes should be done basing on matrix parameters which are described in this work and presented in the results.

The tests were carried out only for one selection of α and β parameters and yet the obtained results were very promising. It seems that further experiments of indexes proposed by *Kułakowski* and *Szybowski* (I_α,$I_{\alpha,\beta}$) should be dedicated to choice of values of parameters α and β in a way that will give even better results. How it was mentioned in this work, these indexes could be a safer alternative to the index K.

An interesting line of enquiry can use PC method for incomplete matrices when somehow a part of comparisons is missing. Such works have already appeared. However, this paper shows that also calculating inconsistency for such matrices is possible. The PC method has been expanding for many years, still discovering new potential and testing new places where it can be applied.