Package 'correspondenceTables'

July 22, 2025

Type Package

Title Creating Correspondence Tables Between Two Statistical Classifications

Date 2022-09-25

Version 0.7.4

Description A candidate correspondence table between two classifications can be created when there are correspondence tables leading from the first classification to the second one via intermediate 'pivot' classifications.
The correspondence table between two statistical classifications can be updated when one of the classifications gets updated to a new version.

License EUPL

Encoding UTF-8

Imports data.table

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

URL https://github.com/eurostat/correspondenceTables

BugReports https://github.com/eurostat/correspondenceTables/issues

Maintainer Mátyás Mészáros <matyas.meszaros@ec.europa.eu>

RoxygenNote 7.1.2

Author Vasilis Chasiotis [aut] (Department of Statistics, Athens University of Economics and Business),
Photis Stavropoulos [aut] (Quantos S.A. Statistics and Information Systems),
Martin Karlberg [aut],
Mátyás Mészáros [cre]

Repository CRAN

Date/Publication 2022-09-27 10:50:02 UTC

Contents

`newCorrespondenceTable`

Ex novo creation of candidate correspondence tables between two classifications via pivot tables

---

Description

Creation of a candidate correspondence table between two classifications, A and B, when there are correspondence tables leading from the first classification to the second one via k intermediate pivot classifications $C_1, \ldots, C_k$. The correspondence tables leading from A to B are A:C_1, $\{C_i$:C_{i+1}: $1 \leq i \leq k-1\}$, B:C_k.

Usage

```
newCorrespondenceTable(
  Tables,
  CSVout = NULL,
  Reference = "none",
  MismatchTolerance = 0.2
)
```

Arguments

`Tables`	A string of type character containing the name of a csv file which contains the names of the files that contain the classifications and the intermediate correspondence tables (see "Details" below).
`CSVout`	The preferred name for the *output csv files* that will contain the candidate correspondence table and information about the classifications involved. The valid values are `NULL` or strings of type `character`. If the selected value is `NULL`, the default, no output file is produced. If the value is a string, then the output is exported into two csv files whose names contain the provided name (see "Value" below).
`Reference`	The reference classification among A and B. If a classification is the reference to the other, and hence *hierarchically superior* to it, each code of the other classification is expected to be mapped to at most one code of the reference classification. The valid values are `"none"`, `"A"`, and `"B"`. If the selected value is `"A"` or `"B"`, a "Review" flag column (indicating the records violating this expectation) is included in the output (see "Explanation of the flags" below).
`MismatchTolerance`	The maximum acceptable proportion of rows in the candidate correspondence table which contain no code for classification A or no code for classification B. The default value is `0.2`. The valid values are real numbers in the interval [0, 1].

Details

File and file name requirements:

- The file that corresponds to argument `Tables` and the files to which the contents of `Tables` lead, must be in *csv format with comma as delimiter*. If full paths are not provided, then these files must be available in the working directory. No two filenames provided must be identical.
- The file that corresponds to argument `Tables` must contain filenames, *and nothing else*, in a $(k+2) \times (k+2)$ table, where k, a positive integer, is the number of "pivot" classifications. The cells in the main diagonal of the table provide the filenames of the files which contain, with this order, the classifications A, C_1, ..., C_k and B. The off-diagonal directly above the main diagonal contains the filenames of the files that contain, with this order, the correspondence tables A:C_1, {C_i:C_{i+1}, $1 \leq i \leq k-1$} and B:C_k. All other cells of the table must be empty.
- If any of the two files where the output will be stored is read protected (for instance because it is open elsewhere) an error message will be reported and execution will be halted.

Classification table requirements:

- Each of the files that contain classifications must contain at least one column and at least two rows. The first column contains the codes of the respective classification. The first row contains column headers. The header of the first column is the name of the respective classification (e.g., "CN 2021").
- The classification codes contained in a classification file (expected in its first column as mentioned above) must be unique. No two identical codes are allowed in the column.
- If any of the files that contain classifications has additional columns the first one of them is assumed to contain the labels of the respective classification codes.

Correspondence table requirements:

- The files that contain correspondence tables must contain at least two columns and at least two rows. The first column of the file that contains A:C_1 contains the codes of classification A. The second column contains the codes of classification C_1. Similar requirements apply to the files that contain C_i:C_{i+1}, $1 \leq i \leq k-1$ and B:C_k. The first row of each of the files that contain correspondence tables contains column headers. The names of the first two columns are the names of the respective classifications.
- The pairs of classification codes contained in a correspondence table file (expected in its first two columns as mentioned above) must be unique. No two identical pairs of codes are allowed in the first two columns.

Interdependency requirements:

- At least one code of classification A must appear in both the file of classification A and the file of correspondence table A:C_1.
- At least one code of classification B must appear in both the file of classification B and the file of correspondence table B:C_k, where k, $k \geq 1$, is the number of pivot classifications.
- If there is only one pivot classification, C_1, at least one code of it must appear in both the file of correspondence table A:C_1 and the file of correspondence table B:C_1.

- If the pivot classifications are k with $k \geq 2$ then at least one code of C_1 must appear in both the file of correspondence table A:C_1 and the file of correspondence table C_1:C_2, at least one code of each of the C_i, $i = 2, \ldots, k-1$ (if $k \geq 3$) must appear in both the file of correspondence table C_{i-1}:C_i and the file of correspondence table C_i:C_{i+1}, and at least one code of C_k must appear in both the file of correspondence table C_{k-1}:C_k and the file of correspondence table B:C_k.

Mismatch tolerance:

- The ratio that is compared with `MismatchTolerance` has as numerator the number of rows in the candidate correspondence table which contain no code for classification A or no code for classification B and as denominator the total number of rows of this table. If the ratio exceeds `MismatchTolerance` the execution of the function is halted.

If any of the conditions required from the arguments is violated an error message is produced and execution is stopped.

Value

`newCorrespondenceTable()` returns a list with two elements, both of which are data frames.

- The first element is the candidate correspondence table A:B, including the codes of all "pivot" classifications, augmented with flags "Review" (if applicable), "Redundancy", "Unmatched", "NoMatchFromA", "NoMatchFromB" and with all the additional columns of the classification and intermediate correspondence table files.
- The second element contains the names of classification A, the "pivot" classifications and classification B as read from the top left-hand side cell of the respective input files.
- If the value of argument `CSVout` a string of type `character`, the elements of the list are exported into files of csv format. The name of the file for the first element is the value of argument `CSVout` and the name of the file for the second element is classificationNames_`CSVout`. For example, if `CSVout` = "newCorrespondenceTable.csv", the elements of the list are exported into "newCorrespondenceTable.csv" and "classificationNames_newCorrespondenceTable.csv" respectively.

Explanation of the flags

- The "Review" flag is produced only if argument Reference has been set equal to `"A"` or `"B"`. For each row of the candidate correspondence table, if `Reference` = `"A"` the value of "Review" is equal to `1` if the code of B maps to more than one code of A, and `0` otherwise. If `Reference` = `"B"` the value of "Review" is equal to `1` if the code of A maps to more than one code of B, and `0` otherwise. The value of the flag is empty if the row does not contain a code of A or a code of B.
- For each row of the candidate correspondence table, the value of "Redundancy" is equal to `1` if the row contains a combination of codes of A and B that also appears in at least one other row of the candidate correspondence table.
- For each row of the candidate correspondence table, the value of "Unmatched" is equal to `1` if the row contains a code of A but no code of B or if it contains a code of B but no code of A. The value of the flag is `0` if the row contains codes for both A and B.

- For each row of the candidate correspondence table, the value of "NoMatchFromA" is equal to 1 if the row contains a code of A that appears in the table of classification A but not in correspondence table A:C_1. The value of the flag is 0 if the row contains a code of A that appears in both the table of classification A and correspondencetable A:C_1. Finally, the value of the flag is empty if the row contains no code of A or if it contains a code of A that appears in correspondence table A:C_1 but not in the table of classification A.
- For each row of the candidate correspondence table, the value of "NoMatchFromB" is equal to 1 if the row contains a code of B that appears in the table of classification B but not in correspondence table B:C_k. The value of the flag is 0 if the row contains a code of B that appears in both the table of classification B and correspondence table B:C_k. Finally, the value of the flag is empty if the row contains no code of B or if it contains a code of B that appears in correspondence table B:C_k but not in the table of classification B.

Sample datasets included in the package

Running `browseVignettes("correspondenceTables")` in the console opens an html page in the user's default browser. Selecting HTML from the menu, users can read information about the use of the sample datasets that are included in the package. If they wish to access the csv files with the sample data, users have two options:

- Option 1: Unpack into any folder of their choice the tar.gz file into which the package has arrived. All sample datasets may be found in the "inst/extdata" subfolder of this folder.
- Option 2: Go to the "extdata" subfolder of the folder in which the package has been installed in their PC's R library. All sample datasets may be found there.

Examples

```
{
   ## Application of function newCorrespondenceTable() with "example.csv" being the file
   ## that includes the names the files  and the intermediate tables in a sparse square
  ## matrix containing the 100 rows of the classifications (from ISIC v4 to CPA v2.1 through
   ## CPC v2.1). The desired name for the csv file that will contain the candidate
  ## correspondence table is "newCorrespondenceTable.csv", the reference classification is
   ## ISIC v4 ("A") and the maximum acceptable proportion of unmatched codes between
  ## ISIC v4 and CPC v2.1 is 0.56 (this is the minimum mismatch tolerance for the first 100 row
   ## as 55.5% of the code of ISIC v4 is unmatched).

     tmp_dir<-tempdir()
   A <- read.csv(system.file("extdata", "example.csv", package = "correspondenceTables"),
                 header = FALSE,
                 sep = ",")
    for (i in 1:nrow(A)) {
      for (j in 1:ncol(A)) {
        if (A[i,j]!="") {
          A[i, j] <- system.file("extdata", A[i, j], package = "correspondenceTables")
      }}}
    write.table(x = A,
                file = file.path(tmp_dir,"example.csv"),
                row.names = FALSE,
                col.names = FALSE,
                sep = ",")
```

```
NCT<-newCorrespondenceTable(file.path(tmp_dir,"example.csv"),
                            file.path(tmp_dir,"newCorrespondenceTable.csv"),
                            "A",
                            0.56)

summary(NCT)
head(NCT$newCorrespondenceTable)
NCT$classificationNames
csv_files<-list.files(tmp_dir, pattern = ".csv")
unlink(csv_files)
}
```

`updateCorrespondenceTable` *Update the correspondence table between statistical classifications A and B when A has been updated to version A**

Description

Update the correspondence table between statistical classifications A and B when A has been updated to version A*.

Usage

```
updateCorrespondenceTable(
  A,
  B,
  AStar,
  AB,
  AAStar,
  CSVout = NULL,
  Reference = "none",
  MismatchToleranceB = 0.2,
  MismatchToleranceAStar = 0.2
)
```

Arguments

`A`	A string of the type `character` containing the name of a csv file that contains the original classification A.
`B`	A string of the type `character` containing the name of a csv file that contains classification B.
`AStar`	A string of the type `character` containing the name of a csv file that contains the updated version A*.
`AB`	A string of the type `character` containing the name of a csv file that contains the previous correspondence table A:B.

`AAStar` A string of the type character containing the name of a csv file that contains the *concordance table* A:A*, which contains the mapping between the codes of the two versions of the classification.

`CSVout` The preferred name for the *output csv files* that will contain the updated correspondence table and information about the classifications involved. The valid values are `NULL` or strings of type `character`. If the selected value is `NULL`, the default, no output file is produced. If the value is a string, then the output is exported into two csv files whose names contain the provided name (see "Value" below).

`Reference` The reference classification among A and B. If a classification is the reference to the other, and hence *hierarchically superior* to it, each code of the other classification is expected to be mapped to at most one code of the reference classification. The valid values are `"none"`, `"A"`, and `"B"`. If the selected value is `"A"` or `"B"`, a "Review" flag column is included in the output (see "Explanation of the flags" below).

`MismatchToleranceB`
The maximum acceptable proportion of rows in the updated correspondence table which contain no code of the target classification B, among those which contain a code of A, of A*, or of both. The default value is `0.2`. The valid values are real numbers in the interval [0, 1].

`MismatchToleranceAStar`
The maximum acceptable proportion of rows in the updated correspondence table which contain no code of the updated classification A*, among those which contain a code of A, of B, or of both. The default value is `0.2`. The valid values are real numbers in the interval [0, 1].

Details

File and file name requirements:

- The files that correspond to arguments `A`, `B`, `AStar`, `AB`, `AAStar` must be in *csv format with comma as delimiter*. If full paths are not provided, then these files must be available in the working directory. No two filenames provided must be identical.
- If any of the two files where the output will be stored is read protected (for instance because it is open elsewhere) an error message will be reported and execution will be halted.

Classification table requirements:

- The files that correspond to arguments `A`, `B` and `AStar` must contain at least one column and at least two rows. The first column contains the codes of the respective classification. The first row contains column headers. The name of the first column is the name of the respective classification (e.g., "CN 2021").
- The classification codes contained in a classification file (expected in its first column as mentioned above) must be unique. No two identical codes are allowed in the column.
- If any of the files that correspond to arguments `A`, `B` and `AStar` has additional columns the first one of them is considered as containing the labels of the respective classification codes.

Correspondence and concordance table requirements:

- The files that correspond to arguments AB and AAStar must contain at least two columns and at least two rows. The first column of the file that corresponds to AB contains the codes of classification A. The second column contains the codes of classification B. Similar requirements apply to the file that corresponds to AAStar. The first row of each of these files contains column headers. The names of the first two columns are the names of the respective classifications.
- The pairs of classification codes contained in the concordance and the correspondence table files (expected in their first two columns as mentioned above) must be unique. No two identical pairs of codes are allowed in the first two columns.

Interdependency requirements:

- At least one code of classification A must appear in both the file of concordance table A:A* and the file of correspondence table A:B.
- At least one code of classification A* must appear in both the file of classification A* and the file of concordance table A:A*.
- At least one code of classification B must appear in both the file of classification B and the file of correspondence table A:B.

Mismatch tolerance:

- The ratio that is compared with MismatchToleranceB has as numerator the number of rows of the updated correspondence table which contain a code for A, for A*, or for both, but no code for B and as denominator the number of rows which contain a code for A, for A*, or for both (regardless of whether there is a code for B or not). If the ratio exceeds MismatchToleranceB the execution of the function is halted.
- The ratio that is compared with MismatchToleranceAStar has as numerator the number of rows of the updated correspondence table which contain a code for A, for B, or for both, but no code for A* and as denominator the number of rows which contain a code for A, for B*, or for both (regardless of whether there is a code for A* or not). If the ratio exceeds MismatchToleranceAStar the execution of the function is halted.

If any of the conditions required from the arguments is violated an error message is produced and execution is stopped.

Value

updateCorrespondenceTable() returns a list with two elements, both of which are data frames.

- The first element is the updated correspondence table A*:B augmented with flags "CodeChange", "Review" (if applicable), "Redundancy", "NoMatchToAStar", "NoMatchToB", "NoMatchFromAStar", "NoMatchFromB", "LabelChange", and with all the additional columns of the A, B, AStar, AB and AAStar files.
- The second element contains the names of the original classification A, the target classification B, and the updated version A*, as read from the top left-hand side cell of the respective input files.
- If the value of argument CSVout is a string of type character, the elements of the list are exported into files of csv format. The name of the file for the first element is the value of argument CSVout and the name of the file for the second element is classificationNames_CSVout.

For example, if `CSVout = "updateCorrespondenceTable.csv"`, the elements of the list are exported into "updateCorrespondenceTable.csv" and "classificationNames_updateCorrespondenceTable.csv", respectively.

Explanation of the flags

- For each row of the updated correspondence table, the value of "CodeChange" is equal to `1` if the code of A contained in this row maps -in this or any other row of the table- to a different code of A*, and `0` otherwise. The value of "CodeChange" is empty if either the code of A, or the code of A*, or both are missing.
- The "Review" flag is produced only if argument `Reference` has been set equal to `"A"` or `"B"`. For each row of the updated correspondence table, if `Reference = "A"` the value of "Review" is equal to `1` if the code of B maps to more than one code of A*, and `0` otherwise. If `Reference = "B"` the value of "Review" is equal to `1` if the code of A* maps to more than one code of B, and `0` otherwise. The value of the flag is empty if either the code of A*, or the code of B, or both are missing.
- For each row of the updated correspondence table, the value of "Redundancy" is equal to `1` if the row contains a combination of codes of A* and B that also appears in at least one other row of the updated correspondence table. The value of the flag is empty if both the code of A* and the code of B are missing.
- For each row of the updated correspondence table, the value of "NoMatchToAStar" is equal to `1` if there is a code for A, for B, or for both, but no code for A*. The value of the flag is `0` if there are codes for both A and A* (regardless of whether there is a code for B or not). Finally, the value of "NoMatchToAStar" is empty if neither A nor B have a code in this row.
- For each row of the updated correspondence table, the value of "NoMatchToB" is equal to `1` if there is a code for A, for A*, or for both, but no code for B. The value of the flag is `0` if there are codes for both A and B (regardless of whether there is a code for A* or not). Finally, the value of "NoMatchToB" is empty if neither A nor A* have a code in this row.
- For each row of the updated correspondence table, the value of "NoMatchFromAStar" is equal to `1` if the row contains a code of A* that appears in the table of classification A* but not in the concordance table A:A*. The value of the flag is `0` if the row contains a code of A* that appears in both the table of classification A* and the concordance table A:A*. Finally, the value of the flag is empty if the row contains no code of A* or if it contains a code of A* that appears in the concordance table A:A* but not in the table of classification A*.
- For each row of the updated correspondence table, the value of "NoMatchFromB" is equal to `1` if the row contains a code of B that appears in the table of classification B but not in the correspondence table A:B. The value of the flag is `0` if the row contains a code of B that appears in both the table of classification B and the correspondence table A:B. Finally, the value of the flag is empty if the row contains no code of B or if it contains a code of B that appears in the correspondence table A:B but not in the table of classification B.
- For each row of the updated correspondence table, the value of "LabelChange" is equal to `1` if the labels of the codes of A and A* are different, and `0` if they are the same. Finally, the value of "LabelChange" is empty if either of the labels, or both labels, are missing. Lower and upper case are considered the same, and punctuation characters are ignored when comparing code labels.

Sample datasets included in the package

Running `browseVignettes("correspondenceTables")` in the console opens an html page in the user's default browser. Selecting HTML from the menu, users can read information about the use of the sample datasets that are included in the package. If they wish to access the csv files with the sample data, users have two options:

- Option 1: Unpack into any folder of their choice the tar.gz file into which the package has arrived. All sample datasets may be found in the "inst/extdata" subfolder of this folder.
- Option 2: Go to the "extdata" subfolder of the folder in which the package has been installed in their PC's `R` library. All sample datasets may be found there.

Examples

```
{
## Application of function updateCorrespondenceTable() with NAICS 2017 being the
## original classification A, NACE being the target classification B, NAICS 2022
## being the updated version A*, NAICS 2017:NACE being the previous correspondence
## table A:B, and NAICS 2017:NAICS 2022 being the A:A* concordance table. The desired
## name for the csv file that will contain the updated correspondence table is
## "updateCorrespondenceTable.csv", there is no reference classification, and the
## maximum acceptable proportions of unmatched codes between the original
## classification A and the target classification B, and between the original
## classification A and the updated classification A* are 0.5 and 0.3, respectively.

tmp_dir<-tempdir()
A <- system.file("extdata", "NAICS2017.csv", package = "correspondenceTables")
AStar <- system.file("extdata", "NAICS2022.csv", package = "correspondenceTables")
B <- system.file("extdata", "NACE.csv", package = "correspondenceTables")
AB <- system.file("extdata", "NAICS2017_NACE.csv", package = "correspondenceTables")
AAStar <- system.file("extdata", "NAICS2017_NAICS2022.csv", package = "correspondenceTables")

UPC <- updateCorrespondenceTable(A,
                                 B,
                                 AStar,
                                 AB,
                                 AAStar,
                                 file.path(tmp_dir,"updateCorrespondenceTable.csv"),
                                 "none",
                                 0.5,
                                 0.3)

summary(UPC)
head(UPC$updateCorrespondenceTable)
UPC$classificationNames
csv_files<-list.files(tmp_dir, pattern = ".csv")
if (length(csv_files)>0) unlink(csv_files)
   }
```

Index