Journal of Statistical Software

August 2018, Volume 86, Code Snippet 2. *doi: 10.18637/jss.v086.c02*

mipfp: An R Package for Multidimensional Array Fitting and Simulating Multivariate Bernoulli Distributions

Johan Barthélemy
University of Wollongong

Thomas Suesse
University of Wollongong

Abstract

This paper explains the **mipfp** package for R with the core functionality of updating an d-dimensional array with respect to given target marginal distributions, which in turn can be multi-dimensional. The implemented methods include the iterative proportional fitting procedure (IPFP), the maximum likelihood method, the minimum chi-square and least squares procedures. The package also provides an application of the IPFP to simulate data from a multivariate Bernoulli distribution. The functionalities of the package are illustrated through two practical examples: the update of a 3-dimensional contingency table to match the targets for a synthetic population and the estimation and simulation of the joint distribution of the binary attribute *impaired pulmonary function* as used by Qaqish, Zink, and Preisser (2012).

Keywords: iterative proportional fitting procedure, maximum likelihood, minimum chi-square, minimum least squares, multivariate Bernoulli distributions, R.

1. Introduction and motivation

Combining information from two or more data sets is an operation commonly required to estimate unknown population counts. Typically this involves integrating fully detailed and disaggregated data from one source with aggregated data from another source. Such processes are frequently implemented when generating synthetic populations (Beckman, Baggerly, and McKay 1996; Guo and Bhat 2007; Barthélemy and Cornelis 2012; Huynh, Barthélemy, and Perez 2016)

For example, let x_{ijk} be the unknown population count referring to a cell of a 3-way contingency table. Index i stands for the household type, j for the gender and k for the age category. Often only marginal tables are released by statistical agencies, such as the gender

population counts $x_{\bullet j \bullet} = \sum_i \sum_k x_{ijk}$. A survey, a random sample from a population, or previous estimates may also be available to generate an initial table x^*_{ijk} known as the *seed*. The aim is, then, to estimate x_{ijk} from the marginal counts and the seed.

A number of packages already exist in R (R Core Team 2018) to achieve estimation using the iterative proportional fitting algorithm (IPFP), including **ipfp** (Blocker 2016), but they are usually limited to initial 2-dimensional tables and 1-dimensional target margins. These packages, however, do not provide a facility to assess the variability of the estimators.

The R function `loglin` from the **stats** package (R Core Team 2018) also relies on the IPFP to compute maximum likelihood estimates for log-linear models of multidimensional contingency tables. However this function only relies on the margins of the given initial array to perform the fitting and does not allow the use of externally supplied margins. The fitted log-linear models also assume a model of conditional independence given the margins, which can be an unrealistic assumption. Similar functionality is also provided by the packages **MASS** (Venables and Ripley 2002) and **cat** (Harding and Tusell 2012).

These observations led the authors to create the **mipfp** package (Barthélemy and Suesse 2018) for R providing in its original release a user friendly multidimensional implementation of the IPFP (giving rise to the name of the package). The current version 3.2.1 of the package also includes several other distance-based fitting methods such as: maximum likelihood (ML), minimum χ^2 (MCSQ) and minimum least square (LSQ). All methods can deal with contingency tables and target margins of arbitrary dimensions. The package also includes an application of the IPFP to simulate data from multivariate Bernoulli distributions, as well as estimating their parameters from given marginal probabilities and association parameters, such as correlations and odds ratios. The package is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=mipfp`.

The remainder of this paper is organized as follows. In Section 2, we first present the fitting methods available in the package before illustrating how they are used. The simulation and estimation of a multivariate Bernoulli distribution using the package functions is detailed in Section 3. Concluding remarks are found in Section 4.

2. Fitting multidimensional arrays

In this section we will briefly detail the methods implemented in the **mipfp** package to fit an initial multi-way (contingency) table with respect to known target marginal distributions (counts). The interested reader can find more details in Suesse, Namazi-Rad, Mokhtarian, and Barthélemy (2017).

The notations in this section refer to a 3-way table, which can be straightforwardly generalized to any dimensions. Assume three categorical variables X_1, X_2 and X_3 with I, J and K levels, respectively. Their initial contingency table, referred to as the *seed*, has given initial components (or cell counts) $x^*_{ijk} \in \mathbb{R}^+$ where $i \in \{1, \ldots, I\}$, $j \in \{1, \ldots, J\}$ and $k \in \{1, \ldots, K\}$ correspond to the level of the first, the second and the third variable, respectively. The unknown and estimated components of the fitted table are denoted by x_{ijk} and $\hat{x}_{ijk} \in \mathbb{R}^+$, respectively. The notations $\mathbf{x}$, $\mathbf{x}^*$ and $\hat{\mathbf{x}}$ represent the vectors containing the elements of the unknown, initial and fitted tables respectively. The vectorization (or reshaping) of the multidimensional arrays is performed with the last index changing fastest. The set of known

desired target marginal counts $\mathcal{M}$ is a non-empty subset of

$$\mathcal{T} = \{(x_{\bullet jk}), (x_{i\bullet k}), (x_{ij\bullet}), (x_{\bullet\bullet k}), (x_{\bullet j\bullet}), (x_{i\bullet\bullet})\ \forall i, j, k\}\,,$$

where $\bullet$ refers to the summation over the corresponding variable, e.g., $x_{\bullet jk} = \sum_i x_{ijk}$. It should be noted that in this particular case each target is either a vector or a matrix. It is also assumed that each component of the margins is non-negative. Consistency across the target margins is also assumed, i.e., they all sum up to N. For example:

$$N = x_{\bullet\bullet\bullet} = \sum_{jk} x_{\bullet jk} = \sum_{ik} x_{i\bullet k} = \sum_{ij} x_{ij\bullet} = \sum_{k} x_{\bullet\bullet k} = \sum_{j} x_{\bullet j\bullet} = \sum_{i} x_{i\bullet\bullet}. \tag{1}$$

The initial cell probabilities are given by the sample proportions $\pi^*_{ijk} = x^*_{ijk}/N^*$ where $N^* = x^*_{\bullet\bullet\bullet}$. Similarly the estimated cell probabilities are obtained by $\hat{\pi}_{ijk} = \hat{x}_{ijk}/N$. Vectorization of the unknown, initial and estimated cell probabilities are denoted by $\boldsymbol{\pi}$, $\boldsymbol{\pi}^*$ and $\hat{\boldsymbol{\pi}}$. Finally let $\mathbf{A}$ be the matrix of full rank such that

$$\mathbf{A}^\top \boldsymbol{\pi} = \begin{pmatrix} \mathbf{m} \\ 1 \end{pmatrix},$$

where vector $\mathbf{m}$ contains all components but one (to insure that $\mathbf{A}$ has a full rank) of every target in $\mathcal{M}$ divided by N, i.e., the target probability margins. Note that $\mathbf{A}$ has $r = I+J+K$ rows and c columns (the last one being a vector of ones), and the degrees of freedom of the $\hat{x}_{ijk}$ is then $r - c$.

The aim is to find a suitable estimator $\hat{x}_{ijk}$ for x_{ijk}, such that every resulting margin equals its corresponding target margin in set $\mathcal{M}$. This can be achieved by using either the IPFP, the maximum likelihood (ML) method, the minimum χ^2 estimation (CHISQ) method or the least square estimation method (LSQ). These methods are briefly detailed in the remainder of this section, followed by illustrating their application using the package's functions on real world data.

2.1. Iterative proportional fitting procedure

One of the most popular approaches to estimate a d-way table based on known marginal tables and an initial contingency table is the iterative proportional fitting procedure originally described by Deming and Stephan (1940), which has been extensively studied over the decades (Lovelace, Birkin, Ballas, and van Leeuwen 2015). This procedure is also known in the literature as *raking*, *matrix scaling* or the *RAS algorithm*.

This method is formally described below when $d = 3$ but can easily be extended to any number of dimension. For the sake of completeness we also assume that every target margin is available, i.e., $\mathcal{M} = \mathcal{T}$. It should be noted that this assumption is not always satisfied with real world data. The procedure iteratively updates the cells of the array depending on the targets. The adjustments at iteration l are computed by the equations:

$$\begin{aligned}
x^{(1)}_{ijk} &= x^{l-1}_{ijk} . \frac{x_{\bullet jk}}{x^{l-1}_{\bullet jk}} \quad \forall i,j,k; & x^{(2)}_{ijk} &= x^{(1)}_{ijk} . \frac{x_{i\bullet k}}{x^{(1)}_{i\bullet k}} \quad \forall i,j,k; \\
x^{(3)}_{ijk} &= x^{(2)}_{ijk} . \frac{x_{ik\bullet}}{x^{(2)}_{ij\bullet}} \quad \forall i,j,k; & x^{(4)}_{ijk} &= x^{(3)}_{ijk} . \frac{x_{\bullet\bullet k}}{x^{(3)}_{\bullet\bullet k}} \quad \forall i,j,k; \\
x^{(5)}_{ijk} &= x^{(4)}_{ijk} . \frac{x_{\bullet j\bullet}}{x^{(4)}_{\bullet j\bullet}} \quad \forall i,j,k; & x^{l}_{ijk} &= x^{(5)}_{ijk} . \frac{x_{i\bullet\bullet}}{x^{(5)}_{i\bullet\bullet}} \quad \forall i,j,k.
\end{aligned} \tag{2}$$

These iterations are performed until either a maximum number of iterations `iter` has been performed or the following stopping criterion is reached:

$$\max |x_{ijk}^{l-1} - x_{ijk}^{l}| \leq \texttt{tol} \quad \forall i, j, k,$$

where $\texttt{tol} \in \mathbb{R}_o^+$ is a small constant. By default $\texttt{tol} = 10^{-11}$ and $\texttt{iter} = 1000$ in the current implementation. The values at the last iteration are the $\hat{x}_{ijk}$.

If only a subset of the target margins is available, i.e., $\mathcal{M} \subset \mathcal{T}$, then only the relevant adjustments in (2) are performed.

Finally it can be noted that if an initial table is not available, this procedure can still be used by choosing initial values for all x_{ijk}^0. The choice of those initial values must then be done carefully for the following reasons:

- IPFP preserves the correlation structure defined by the odds ratio of the initial table (Mosteller 1968).
- If the initial table contains some cells $x_{ijk}^0 = 0$, then it can be easily observed that the value of cells will remain 0 across the iterations. This property allows to easily fit tables with structural zeros.
- If $x_{ijk}^0 = 1 \ \forall i, j, k$ then IPFP produces the same results as a log-linear model of conditional independence given the margins (such as the R function `loglin`, among others, would). This may or may not be adequate depending on the application and should be then be done only when such model is deemed adequate.

2.2. Distance-based approaches

As an alternative to the IPFP, one can use a distance-based approach, consisting of solving the following optimization problem:

$$\begin{aligned} \hat{\boldsymbol{\pi}} &= \arg\min_{\boldsymbol{\pi}} f(\boldsymbol{\pi}) \\ \text{s.t.} \quad \mathbf{A}^\top \boldsymbol{\pi} &= \begin{pmatrix} \mathbf{m} \\ 1 \end{pmatrix} \end{aligned}$$

to find estimates $\hat{\pi}_{ijk}$, where different specifications of $f(\boldsymbol{\pi})$ lead to different models, see Little and Wu (1991) for the corresponding models. In particular, we consider the following functions:

- the maximum likelihood (ML) method under random sampling obtained by:

$$f(\boldsymbol{\pi}) = -N^* \sum_{i,j,k} \pi_{ijk}^* \ln(\pi_{ijk});$$

- the minimum χ^2 method (MCSQ) characterized by:

$$f(\boldsymbol{\pi}) = \sum_{i,j,k} \frac{\left(\pi_{ijk} - \pi_{ijk}^*\right)^2}{\pi_{ijk}};$$

	IPFP	ML	LSQ	MCSQ
$\mathbf{D}_1$	$\hat{\pi}$	$\hat{\pi}^2/\pi^*$	π^*	$\hat{\pi}^4/(\pi^*)^3$
$\mathbf{D}_2$	π^*	$\hat{\pi}^2/\pi^*$	$(\pi^*)^3/\hat{\pi}^2$	$\hat{\pi}^4/(\pi^*)^3$

Table 1: Diagonal elements of $\mathbf{D}_1$ and $\mathbf{D}_2$. The vector division is performed component-wise.

- and the minimum least square (LSQ) method defined by:

$$f(\boldsymbol{\pi}) = \sum_{i,j,k} \frac{\left(\pi_{ijk} - \pi^*_{ijk}\right)^2}{\pi^*_{ijk}}.$$

The count estimates are finally obtained by the relation $\hat{x}_{ijk} = N\hat{\pi}_{ijk}$.

Interestingly, all four estimations methods (IPFP, ML, MCSQ and LSQ) are also maximum likelihood methods for various mis-specification models. Such models link a target population with known margins and a sample obtained from a different population, the so-called non-target population. This is practically important, as often the available sample is not a real sample from the target population and a mis-specification model then allows estimation of the target population counts (see Little and Wu 1991; Suesse *et al.* 2017, for more details).

2.3. Covariance estimation

Assuming that the distribution of the random sample used to derive π^* can be approximated by a multinomial distribution[1], then following Little and Wu (1991) and Freeman and Koch (1976) the asymptotic covariance matrix of the estimated probabilities providing an uncertainty measure is derived from the Delta method and formally defined as:

$$\widehat{\mathsf{COV}}(\hat{\boldsymbol{\pi}}) = \frac{1}{N^*}\mathbf{U}(\mathbf{U}^\top\mathbf{D}_1^{-1}\mathbf{U})^{-1}(\mathbf{U}^\top\mathbf{D}_2^{-1}\mathbf{U})(\mathbf{U}^\top\mathbf{D}_1^{-1}\mathbf{U})^{-1}\mathbf{U}^\top,$$

where $\mathbf{D}_1$ and $\mathbf{D}_2$ are diagonal matrices whose diagonal elements are given in Table 1 and the matrix $\mathbf{U}$ is the orthogonal complement of $\mathbf{A}$ such that $\mathbf{A}^\top\mathbf{U} = \mathbf{0}$ and $(\mathbf{A}|\mathbf{U})$ has full rank.

The asymptotic covariance of the estimated cell counts is then simply given by:

$$\widehat{\mathsf{COV}}(\hat{\mathbf{x}}) = N^2\widehat{\mathsf{COV}}(\hat{\boldsymbol{\pi}}).$$

Lang (2004, 2005) also proposed the following covariance matrix for the estimators:

$$\widehat{\mathsf{COV}}(\hat{\boldsymbol{\pi}}) = \frac{1}{N^*}\left(\mathbf{D}_{\hat{\boldsymbol{\pi}}} - \hat{\boldsymbol{\pi}}\hat{\boldsymbol{\pi}}^\top - \mathbf{D}_{\hat{\boldsymbol{\pi}}}\mathbf{H}_{\hat{\boldsymbol{\pi}}}(\mathbf{H}_{\hat{\boldsymbol{\pi}}}^\top\mathbf{D}_{\hat{\boldsymbol{\pi}}}\mathbf{H}_{\hat{\boldsymbol{\pi}}})^{-1}\mathbf{H}_{\hat{\boldsymbol{\pi}}}^\top\mathbf{D}_{\hat{\boldsymbol{\pi}}}\right),$$

where $\mathbf{H}_{\hat{\boldsymbol{\pi}}}$ denotes $\mathbf{J}_\mathbf{h}(\hat{\boldsymbol{\pi}})$, the Jacobian evaluated in $\hat{\boldsymbol{\pi}}$ of the function:

$$\mathbf{h}(\mathbf{p}) = \mathbf{A}_{-1}^T\mathbf{p} - \mathbf{m}.$$

The matrix $\mathbf{A}_{-1}$ is different from $\mathbf{A}$ in the sense that the last column of ones is removed.

[1] Typically the target population size is larger than the sample size hence this assumption is not unrealistic.

Variable	Values
Household type	C (couple), F (family with children), I (isolated), N (non family)
Gender	F (female); H (male)
Professional status	A (active); E (student); I (inactive)
Education level	O (none); P (primary); S (high school); U (higher education)
Driving license	O (no); P (yes)
Age class	0 (0–5); 1 (6–17); 2 (19–39); 3 (40–59); 4 (60+)

Table 2: Individuals' characteristics.

2.4. Goodness of fit statistics

Lang (2004) proposes the following three statistics to perform a test of the null hypothesis, informally expressed as:

$$H_0 \ : \ \text{data } x^*_{ijk} \text{ agree with } \mathcal{M} \qquad \text{vs.} \qquad H_1 \ : \ \text{data } x^*_{ijk} \text{ do not agree with } \mathcal{M},$$

and formally as:

$$H_0 \ : \ \mathbf{h}(\boldsymbol{\pi}^*) = \mathbf{0} \qquad \text{vs.} \qquad H_1 \ : \ \mathbf{h}(\boldsymbol{\pi}^*) \neq \mathbf{0}.$$

The statistics are:

- the Wilk's log-likelihood ratio statistic:

$$G^2 = 2 \sum_{ijk} x^*_{ijk} \ln \left(\frac{\pi^*_{ijk}}{\hat{\pi}_{ijk}} \right);$$

- the Wald statistic:

$$W^2 = \mathbf{h}(\mathbf{x}^*)^\top \left(\mathbf{H}^\top_{\mathbf{x}^*} \mathbf{D}_{\mathbf{x}^*} \mathbf{H}_{\mathbf{x}^*} \right)^{-1} \mathbf{h}(\mathbf{x}^*);$$

- and the Pearson χ^2 statistic:

$$\chi^2 = (\mathbf{x}^* - n\hat{\boldsymbol{\pi}})^\top \mathbf{D}^{-1}_{n\hat{\boldsymbol{\pi}}} (\mathbf{x}^* - n\hat{\boldsymbol{\pi}}).$$

The degrees of freedom for these statistics corresponds to the number of components in $\mathbf{m}$.

2.5. Functions description and illustrative example

The IPFP and the distance-based approaches are implemented in the `Estimate()` function. Its arguments, along with their description, are listed in Table 4. The minimal requirements for `Estimate()` consist of an initial array to be updated, a list describing the dimensions of the target margins, a list containing the data of the target margins and the method to be used for the estimation. The function will return an object of class '`mipfp`', as detailed in Table 5 containing the updated array and information about the convergence of the selected algorithm. It should be noted that in the case of the distance-based approaches, the optimization is performed using the function `solnp` from the package **Rsolnp** (Ghalanos and Theussl 2015).

To illustrate the functionalities of the package, the data frame `spnamur` representing a synthetic population of 105,248 individuals for the city Namur (Belgium) is used. We refer the

interested reader to Barthélemy and Toint (2013) for a detailed analysis and description of the generation of this data set. The variables of the data set are detailed in Table 2. For the sake of simplicity we disregard in this example the last three variables, and focus only on *household type* (*hht*), *gender* (*gen*) and *professional status* (*pro*).

For illustration purposes we obtain the seed and the target margins from the synthetic population. The margins are extracted from the contingency table of the synthetic population. The seed is obtained by taking a 10% simple random sample extracted (without replacement) from the synthetic population.

Nevertheless it should be noted that in a real world application, the original data from which the seed and the target margins are derived are usually not available. Typically the target margins come from current aggregated census data while the seed is provided from other data sources, such as surveys or disaggregated partial census data.

The package and the data are loaded and a random seed is set with:

```
R> library("mipfp")
R> set.seed(1234)
R> data("spnamur", package = "mipfp")
```

The 3-dimensional contingency table $hht \times gen \times pro$ is obtained by:

```
R> spnamur.sub <- subset(spnamur, select = Household.type:Prof.status)
R> true.table <- table(spnamur.sub)
```

The target margins can then be easily extracted from this table. In this illustrative example, we consider the (multi-dimensional) target marginal distribution detailed in Table 3 which is generated by:

```
R> tgt.hht <- apply(true.table, 1, sum)
R> tgt.hht.gen <- apply(true.table, c(1, 2), sum)
R> tgt.gen.pro <- apply(true.table, c(2, 3), sum)
```

It can be observed that the targets defined in `tgt.hht` can be derived from `tgt.hht.gen`. Ideally the former should be discarded, but it is still included in this example to demonstrate that the package functions are able to detect and remove the unnecessary target margins.[2] The target margins and their description (i.e., the index of the variables involved) are then stored in the lists `tgt.list.dims` and `tgt.data` respectively by:

```
R> tgt.data <- list(tgt.hht, tgt.hht.gen, tgt.gen.pro)
R> tgt.list.dims <- list(1, c(1, 2), c(2, 3))
```

The next step extracts the 10% sample from the original synthetic population in order to generate the initial *seed* contingency table `seed.table`:

```
R> sample.idx <- sample(nrow(spnamur), ceiling(nrow(spnamur) * 0.10))
R> seed.df <- spnamur.sub[sample.idx, ]
R> seed.table <- table(seed.df)
```

[2]Hence insuring that $\mathbf{A}$ has full rank, an assumption required for the convergence of the distance-based approach and the computation of the covariance estimators.

Name	Variables	Dimensions
`tgt.hht`	Household type	4
`tgt.hht.gen`	Household type × Gender	4 × 2
`tgt.gen.pro`	Gender × Professional status	2 × 3

Table 3: Description of the target marginal distributions.

Having a seed and a set of target margins, the IPPF and the distance-based approaches can now be applied to obtain an estimate of the true contingency table. This is achieved by using the function `Estimate()` returning an object of class '`mipfp`':

```
R> r.ipfp <- Estimate(seed = seed.table, target.list = tgt.list.dims,
+    target.data = tgt.data,  method = "ipfp")
R> r.ml   <- Estimate(seed = seed.table, target.list = tgt.list.dims,
+    target.data = tgt.data, method = "ml")
R> r.chi2 <- Estimate(seed = seed.table, target.list = tgt.list.dims,
+    target.data = tgt.data, method = "chi2")
R> r.lsq  <- Estimate(seed = seed.table, target.list = tgt.list.dims,
+    target.data = tgt.data, method = "lsq")
```

A summary of the results detailing the values of the estimates, their standard deviation, their t score and associated p value, the absolute maximum deviation between every target and its corresponding generated margin (referred as the margins errors) and the goodness of fit statistics can then be obtained using the `summary()` method. For instance, the summary for the object `r.ipfp` is given by:

```
R> summary(r.ipfp)

Call:
Estimate(seed = seed.table, target.list = tgt.list.dims,
    target.data = tgt.data, method = "ipfp")

Method: ipfp  - convergence: TRUE

       Estimate    StdDev t.value   p.value
C.F.A  3503.915   140.713 24.9011 1.065e-11 ***
F.F.A  6663.454   166.755 39.9596 3.898e-14 ***
I.F.A  3064.765   125.276 24.4641 1.312e-11 ***
N.F.A  2441.866   121.823 20.0444 1.359e-10 ***
C.H.A  5257.993   144.859 36.2973 1.225e-13 ***
F.H.A 11201.509   171.377 65.3619 < 2.2e-16 ***
I.H.A  3363.671   120.921 27.8170 2.880e-12 ***
N.H.A  1722.827   100.330 17.1716 8.206e-10 ***
C.F.E   849.901    85.626  9.9257 3.882e-07 ***
F.F.E  8118.086   140.683 57.7047 4.844e-16 ***
I.F.E   557.856    67.750  8.2340 2.795e-06 ***
N.F.E  2522.157   118.211 21.3360 6.547e-11 ***
```

```
C.H.E   822.065    81.752 10.0556 3.373e-07 ***
F.H.E  8076.191   137.015 58.9438 3.757e-16 ***
I.H.E  1148.761    90.880 12.6404 2.704e-08 ***
N.H.E  2019.983   100.649 20.0695 1.339e-10 ***
C.F.I  7491.184   148.535 50.4339 2.423e-15 ***
F.F.I 11364.461   179.373 63.3566 < 2.2e-16 ***
I.F.I  5108.378   129.340 39.4958 4.480e-14 ***
N.F.I  3310.977   129.792 25.5098 8.011e-12 ***
C.H.I  5736.942   144.312 39.7539 4.145e-14 ***
F.H.I  7012.299   162.893 43.0485 1.604e-14 ***
I.H.I  2339.568   114.756 20.3874 1.115e-10 ***
N.H.I  1549.191    97.144 15.9473 1.926e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Degrees of freedom: 12

Variables:
* Household.type : C F I N
* Gender : F H
* Prof.status : A E I

Margins errors:
       Household.type Household.type.Gender    Gender.Prof.status
         7.275958e-12          3.637979e-12          0.000000e+00

Goodness of fit statistics:
     Stat p.value
G2 10.567  0.4802
W2 10.465  0.4891
X2 10.627  0.4750
Degrees of freedom: 11
```

These first results from the IPFP show that after the convergence of the algorithm, the margins of the estimated cells fit the known target margins. Indeed `Margins errors` reports a small absolute maximum deviation between the desired and estimated margins ($\approx 3.6 \times 10^{-12}$). The p values behind each cell strongly reject the underlying null hypothesis that the cell is zero. Hence every cell is significant in this example.

The goodness of fit statistics defined in Section 2.4 and their associated p value indicate that the seed (10% sample) agrees with the imposed margins, as expected as the sample and the margins were obtained from the same population data. More details about the inputs and outputs of the `summary()` method can be found in Tables 6 and 7. It should be noted that the results of the other methods stored in `r.ml`, `r.chi2` and `r.lsq` are similar to `r.ipfp`.

The confidence intervals of the estimates (Smithson 2002) can be easily computed with the `confint` method as illustrated for the object `r.ipfp` below.

```
R> confint(r.ifpp)
```

```
            2.5%      97.5%
C.F.A  3228.1220  3779.7081
F.F.A  6495.6289  6831.2781
I.F.A  2773.6426  3355.8880
N.F.A  2157.9479  2725.7844
C.H.A  5097.7618  5418.2247
F.H.A 10918.6639 11484.3548
I.H.A  3036.8371  3690.5039
N.H.A  1447.0928  1998.5610
C.F.E   498.3366  1201.4653
F.F.E  7782.1934  8453.9777
I.F.E   289.3115   826.4011
N.F.E  2202.8924  2841.4219
C.H.E   576.5281  1067.6018
F.H.E  7943.4034  8208.9789
I.H.E   895.2599  1402.2627
N.H.E  1782.9809  2256.9843
C.F.I  7313.0616  7669.3064
F.F.I 11139.5439 11589.3779
I.F.I  4869.6100  5347.1468
N.F.I  3079.2871  3542.6662
C.H.I  5482.5537  5991.3300
F.H.I  6815.6559  7208.9429
I.H.I  2142.2988  2536.8376
N.H.I  1358.7915  1739.5896
```

The inputs of the method are documented in Table 8 and the output is a matrix containing the upper and lower bounds of the estimates. Other S3 methods for the objects of class '`mipfp`' are also available such as `vcov()` to compute the variance-covariance matrix of the estimates, `coef()` to extract the estimates and `print()` to print a short description of the object (see Barthélemy and Suesse 2018).

In order to validate the implementation of the different methods, to asses their convergence and to compare their results, the `CompareMaxDev()` function is provided. First we check the absolute maximum deviation between every target and its corresponding generated margin:

```
R> CompareMaxDev(list(r.ipfp, r.ml, r.chi2, r.lsq), echo = TRUE)

Maximum absolute deviation between targets and generated margins:
     Household.type Household.type.Gender Gender.Prof.status
ipfp   7.275958e-12          3.637979e-12       0.000000e+00
ml     2.910383e-11          1.818989e-11       2.182787e-11
chi2   1.455192e-11          1.273293e-11       2.182787e-11
lsq    1.455192e-11          1.273293e-11       1.637090e-11
```

These first results show that after the methods have converged to a solution, the generated margins fit the (true) target margins (regardless of the method), as expected.

In this example the true contingency table is stored in `true.table`. To confirm that the methods performs well we can also compare the estimated contingency table with the true (but usually unknown) table using the same function:

```
R> CompareMaxDev(list(r.ipfp, r.ml, r.chi2, r.lsq), echo = TRUE,
+    true.table = true.table)

Maximum absolute deviation:
     Deviation        Prop
ipfp  227.1840 0.002158559
ml    227.8614 0.002164995
chi2  229.9759 0.002185086
lsq   230.6854 0.002191827
```

The absolute deviation between estimated table counts and the true counts are small, resulting in deviations that are of maximum order of $\approx 0.3\%$.

It should also be noted that if the target margins are not consistent, for instance $\sum_i x_{i\bullet\bullet} = N' \neq N$ as defined in Equation 1, then the seed and target probabilities are automatically used as inputs rather than the counts by the function `Estimate`. In this instance the target becomes $\pi_{i\bullet\bullet} = x_{i\bullet\bullet}/N'$.

Finally the function `Estimate()` can also accommodate missing values (`NA`) in the targets components when the selected method is IPFP.

3. Simulating multivariate Bernoulli distributions

Consider the K binary variables $Y_1, \ldots, Y_K$ with success probabilities $\pi_i = \mathsf{P}(Y_i = 1)$ for $i = 1, \ldots, K$. Under independence, random numbers or data referring to these K variables can easily be generated by generating independently numbers from the Bernoulli distribution. Under dependence simulation of multivariate binary data becomes more complicated, because the underlying distribution is characterized by 2^K probabilities which add up to 1.

When K is large, specifying and determining 2^K probabilities becomes impractical and often infeasible. A simpler approach is provided by using the IPFP, as suggested by various authors, for example Lee (1993) and Gange (1995), and has been applied in various simulation studies, e.g., Bilder, Loughin, and Nettleton (2000); Liu and Suesse (2008); Suesse and Liu (2012) for simulating multivariate binary data. The approach is based on specifying the K probabilities $\pi_1, \ldots, \pi_K$ and the $(K-1) \times K/2$ pairwise-probabilities $\pi_{ij} = \mathsf{P}(Y_i = 1, Y_j = 1)$. The IPFP finds a solution of 2^K probabilities such that the marginal one- and two-dimensional probabilities equal $\{\pi_i\}$ and $\{\pi_{i,j}\}$. In practice, specifying pair-wise probabilities is difficult, as these are bounded by $\{\pi_i\}$, i.e., $\max(0, \pi_i + \pi_j - 1) \leq \pi_{ij} \leq \min(\pi_i, \pi_j)$.

There are often many solutions and the IPFP converges to one of these. There are also other approaches, for example linear programming (Lee 1993), however IPFP is attractive because the final solution has usually strictly positive joint probabilities, meaning that none of the theoretically 2^K sequences can be excluded. This is in contrast to linear programming which often has several zero joint probabilities in the final solution, meaning these sequences can never be generated in the sampling process, an undesirable practical property.

Alternatively correlations and odds ratios are standard measures of association between two random variables and can be used to determine the π_{ij} required by IPFP. While the correlations are frequently used for continuous random variables, the odds ratio is frequently used for categorical variables, because the correlation $\mathsf{COR}(Y_i, Y_j)$ between Y_i and Y_j:

$$\mathsf{COR}(Y_i, Y_j) = \frac{\pi_{ij} - \pi_i \pi_j}{\sqrt{\pi_i(1-\pi_i)\pi_j(1-\pi_j)}}$$

is also bounded in a similar fashion as π_{ij}. In contrast the odds ratio OR_{ij} for variables Y_i and Y_j defined by

$$OR_{ij} = \frac{\mathsf{P}(Y_i = 1, Y_j = 1)\mathsf{P}(Y_i = 0, Y_j = 0)}{\mathsf{P}(Y_i = 0, Y_j = 1)\mathsf{P}(Y_i = 1, Y_j = 0)}$$

is not bounded and can take any value in $(0, \infty)$.

3.1. Function description and illustrative example

Qaqish *et al.* (2012) analyze a frequently used data set on $n = 407$ parents and siblings of subjects with chronic obstructive pulmonary disease and their controls with the binary outcome of interest *impaired pulmonary function*. These data are clustered as observations come from families and family size $K = 1, 2, \ldots, 10$ varies. The primary focus of the authors was to model the probability of impaired pulmonary function as a function of sex, race, age, smoking status and an indicator as to whether a relative had the same disease. While a logit model was applied to model $\{\pi_i\}$, the odds ratio was used as a measure of association. To simplify a model for the $K(K-1)/2$ odds ratios, each distinct family relationship was modeled with a different parameter: parent-parent with α_{PP}, parent-sibling with α_{PS} and sibling-sibling with α_{SS}. Joint estimation of such mean and association models can be achieved with generalized estimating equations (GEE2) or the method of orthogonalized residuals, see Qaqish *et al.* (2012) and Liang, Zeger, and Qaqish (1992) for more information on these approaches. For example OR estimates for the method of orthogonalized residuals, see method $\text{ORTH}_{\text{MOMENT}}$ in Table 3 in Liang *et al.* (1992), are $\alpha_{\text{PP}} = 0.283$, $\alpha_{\text{PS}} = 2.214$, $\alpha_{\text{SS}} = 2.186$. We aim not at estimation of these parameters but at generating data from the underlying joint distribution. Let us consider for simplicity a family of four with two parents and two siblings, then the matrix with the OR's used in this example is loaded by:

```
R> library("mipfp")
R> data("Qaqish", package = "mipfp")
R> or <- Qaqish$or
R> or

         Parent1 Parent2 Sibling1 Sibling2
Parent1      Inf   0.281    2.214    2.214
Parent2    0.281     Inf    2.214    2.214
Sibling1   2.214   2.214      Inf    2.185
Sibling2   2.214   2.214    2.185      Inf
```

and let us fix the $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4)^\top$ for simplicity as:

```
R> p <- c(0.2, 0.4, 0.6, 0.8)
```

which would usually be determined by the logistic regression model for a given set of covariates for the four individuals under consideration. Then estimating the joint distribution via IPFP can be achieved with:

```
R> p.joint <- ObtainMultBinaryDist(odds = or, marg.probs = p)
```

These $2^4 = 16$ joint probabilities are stored in the `p.joint$joint.proba` and can now be used to generate multivariate binary data using `RMultBinary()`. For instance, simulating 100,000 data points of size 4 from the obtained joined-distribution can be done with:

```
R> y.sim <- RMultBinary(n = 1e5, mult.bin.dist = p.joint)$binary.sequences
```

To confirm the results, the estimated probabilities and odds ratios from the generated random data are:

```
R> apply(y.sim, 2, mean)

 Parent1  Parent2 Sibling1 Sibling2
 0.19946  0.40101  0.60081  0.80101

R> cor(y.sim)

            Parent1    Parent2  Sibling1  Sibling2
Parent1   1.0000000 -0.2125456 0.1481007 0.1053726
Parent2  -0.2125456  1.0000000 0.1838986 0.1432878
Sibling1  0.1481007  0.1838986 1.0000000 0.1567362
Sibling2  0.1053726  0.1432878 0.1567362 1.0000000
```

which should be compared with the true correlation matrix which can be obtained from the odds ratios as:

```
R> Odds2Corr(or, p)$corr

            Parent1    Parent2  Sibling1  Sibling2
Parent1   1.0000000 -0.2156821 0.1445775 0.1076353
Parent2  -0.2156821  1.0000000 0.1847014 0.1445775
Sibling1  0.1445775  0.1847014 1.0000000 0.1563619
Sibling2  0.1076353  0.1445775 0.1563619 1.0000000
```

or alternatively the estimated correlation matrix can be converted into OR estimates by:

```
R> Corr2Odds(corr = cor(y.sim), marg.probs = apply(y.sim, 2, mean))$odds

           Parent1   Parent2 Sibling1 Sibling2
Parent1        Inf 0.2875149 2.266229 2.176636
Parent2  0.2875149       Inf 2.205761 2.199028
Sibling1 2.2662294 2.2057614      Inf 2.192198
Sibling2 2.1766357 2.1990276 2.192198      Inf
```

which is approximately the same as the initial OR matrix used in this example.

4. Conclusion

In this paper we have presented the R package **mipfp**. It provides several methods for updating an initial array (or seed) with respect to given target margins, namely the maximum likelihood, minimum χ^2 and minimum least squares methods as well as the well known iterative proportional fitting procedure. The package provides the first and crucial step for generating synthetic populations where a disaggregate sample and aggregate margins are available. Unlike the other fitting methods already implemented in several packages it can also compute an approximation of the covariance matrix and standard errors of the estimates which can be used to assess their accuracy and confidence intervals.

Package **mipfp** also provides an application of the iterative proportional fitting to simulate data from and estimate multivariate Bernoulli distributions, an important application for simulation studies. The main functions of **mipfp** have been successfully illustrated through the use of data sets included in the package.

Extensions will be implemented in future versions of the package. Those include additional uncertainty measures for the estimated cells induced by fixed marginals such as Fréchet bounds (Fienberg 1999; Dobra and Fienberg 2001) and rounding procedures of the estimated counts (Lovelace and Ballas 2013). Finally, we also aim at adding methods for synthetic population generation and other estimation methods.

Acknowledgments

The authors wish to thank Francisco José Goerlich Gisbert, Mohammad-Reza Namazi-Rad, Robin Lovelace and Morgane Dumont for their helpful comments and suggestions as well as for testing thoroughly the package. The authors also wish to gratefully acknowledge the help of Dr. Madeleine Strong Cincotta in the final language editing of this paper.

References

Barthélemy J, Cornelis E (2012). "Synthetic Populations: Review of the Different Approaches." *Technical report*, CEPS/INSTEAD.

Barthélemy J, Suesse T (2018). ***mipfp**: Multidimensional Iterative Proportional Fitting and Alternative Models*. R package version 3.2.1, URL https://CRAN.R-project.org/package=mipfp.

Barthélemy J, Toint PL (2013). "Synthetic Population Generation without a Sample." *Transportation Science*, **47**(2), 266–279. doi:10.1287/trsc.1120.0408.

Beckman RJ, Baggerly KA, McKay MD (1996). "Creating Synthetic Baseline Populations." *Transportation Research Part A: Policy and Practice*, **30**(6), 415–429. doi:10.1016/0965-8564(96)00004-3.

Bilder CR, Loughin TM, Nettleton D (2000). "Multiple Marginal Independence Testing for Pick Any/C Variables." *Communications in Statistics – Simulation and Computation*, **29**(4), 1285–1316. doi:10.1080/03610910008813665.

Blocker AW (2016). ***ipfp**: Fast Implementation of the Iterative Proportional Fitting Procedure in C*. R package version 1.0.1, URL https://CRAN.R-project.org/package=ipfp.

Deming WE, Stephan FF (1940). "On a Least Squares Adjustment of a Sampled Frequency Table When the Expected Marginal Totals Are Known." *The Annals of Mathematical Statistics*, **11**(4), 427–444. doi:10.1214/aoms/1177731829.

Dobra A, Fienberg SE (2001). "Bounds for Cell Entries in Contingency Tables Induced by Fixed Marginal Totals with Applications to Disclosure Limitation." *Statistical Journal of the United Nations Economic Commission for Europe*, **18**(4), 363–371.

Fienberg SE (1999). "Fréchet and Bonferroni Bounds for Multi-Way Tables of Counts with Applications to Disclosure Limitation." In *Statistical Data Protection (SDP'98) Proceedings*, pp. 115–129.

Freeman DH, Koch GG (1976). "An Asymptotic Covariance Structure for Testing Hypotheses on Raked Contingency Tables from Complex Sample Surveys." In *Proccedings of the Social Statistics Section*, pp. 330–335.

Gange SJ (1995). "Generating Multivariate Categorical Variates Using the Iterative Proportional Fitting Algorithm." *The American Statistician*, **49**(2), 134–138. doi:10.2307/2684626.

Ghalanos A, Theussl S (2015). ***Rsolnp**: General Non-Linear Optimization Using Augmented Lagrange Multiplier Method*. R package version 1.16, URL https://CRAN.R-project.org/package=Rsolnp.

Guo Y, Bhat CR (2007). "Population Synthesis for Microsimulating Travel Behavior." *Transportation Research Record: Journal of the Transportation Research Board*, **2014**(1), 92–101. doi:10.3141/2014-12.

Harding T, Tusell F (2012). ***cat**: Analysis of Categorical-Variable Datasets with Missing Values*. R package version 0.0-6.5. Original by Joseph L. Schafer, URL https://CRAN.R-project.org/package=cat.

Huynh N, Barthélemy J, Perez P (2016). "A Heuristic Combinatorial Optimisation Approach to Synthesising a Population for Agent Based Modelling Purposes." *Journal of Artificial Societies and Social Simulation*, **19**(4), 11. doi:10.18564/jasss.3198.

Lang JB (2004). "Multinomial-Poisson Homogeneous Models for Contingency Tables." *The Annals of Statistics*, **32**(1), 340–383. doi:10.1214/aos/1079120140.

Lang JB (2005). "Homogeneous Linear Predictor Models for Contingency Tables." *Journal of the American Statistical Association*, **100**(469), 121–134. doi:10.1198/016214504000001042.

Lee AJ (1993). "Generating Random Binary Deviates Having Fixed Marginal Distributions and Specified Degrees of Association." *The American Statistician*, **47**(3), 209–215. doi:10.1080/00031305.1993.10475980.

Liang KY, Zeger SL, Qaqish B (1992). "Multivariate Regression Analyses for Categorical Data." *Journal of the Royal Statistical Society B*, **54**(1), 3–40.

Little RJ, Wu MM (1991). "Models for Contingency Tables with Known Margins When Target and Sampled Populations Differ." *Journal of the American Statistical Association*, **86**(413), 87–95. doi:10.2307/2289718.

Liu I, Suesse T (2008). "The Analysis of Stratified Multiple Responses." *Biometrical Journal*, **50**(1), 135–149. doi:10.1002/bimj.200710395.

Lovelace R, Ballas D (2013). "'Truncate, Replicate, Sample': A Method for Creating Integer Weights for Spatial Microsimulation." *Computers, Environment and Urban Systems*, **41**, 1–11. doi:10.1016/j.compenvurbsys.2013.03.004.

Lovelace R, Birkin M, Ballas D, van Leeuwen E (2015). "Evaluating the Performance of Iterative Proportional Fitting for Spatial Microsimulation: New Tests for an Established Technique." *Journal of Artificial Societies and Social Simulation*, **18**(2), 21. doi:10.18564/jasss.2768.

Mosteller F (1968). "Association and Estimation in Contingency Tables." *Journal of the American Statistical Association*, **63**(321), 1–28. doi:10.2307/2283825.

Qaqish BF, Zink RC, Preisser JS (2012). "Orthogonalized Residuals for Estimation of Marginally Specified Association Parameters in Multivariate Binary Data." *Scandinavian Journal of Statistics*, **39**(3), 515–527. doi:10.1111/j.1467-9469.2012.00802.x.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Smithson M (2002). "Central Confidence Intervals." In *Confidence Intervals*, volume 140, chapter 3, pp. 18–31. Sage Publications.

Suesse T, Liu I (2012). "Mantel-Haenszel Estimators of Odds Ratios for Stratified Dependent Binomial Data." *Computational Statistics & Data Analysis*, **56**(9), 2705–2717. doi:10.1016/j.csda.2012.02.015.

Suesse T, Namazi-Rad MR, Mokhtarian P, Barthélemy J (2017). "Estimating Cross-Classified Population Counts of Multidimensional Tables: An Application to Regional Australia to Obtain Pseudo-Census Counts." *Journal of Official Statistics*, **33**(4), 1021–1050. doi:10.1515/jos-2017-0048.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York. URL http://www.stats.ox.ac.uk/pub/MASS4.

A. Documentation

Argument	Description
`seed`	An array storing the initial multi-dimensional table to be updated. Each cell must be non-negative.
`target.list`	A list of dimensions of the marginal target constraints in `target.data`. Each component of the list is an array whose cells indicate which dimension the corresponding margin relates to.
`target.data`	A list containing the data of the target margins. Each component of the list is an array storing a margin. The list order must follow the one defined in `target.list`. Note that the cells of the arrays must be non-negative.
`method`	An optional character string indicating which method is to be used to update the seed. This must be one of the strings `"ipfp"` (iterative proportional fitting procedure, the default.), `"ml"` (maximum likelihood), `"chi2"` (minimum chi-squared), or `"lsq"` (least squares).
`keep.input`	A Boolean indicating if `seed`, `target.data` and `target.list` must be saved in the output when set to `TRUE`.
`...`	Additional arguments that can be passed to the functions `Ipfp()` and `ObtainModelEstimates()`. See their respective documentation for more details (Barthélemy and Suesse 2018).

Table 4: List of arguments for the function `Estimate()`.

Name	Description
`x.hat`	Array of the same dimension as `seed` containing the updated cell values and whose margins match the ones specified in `target.list`.
`p.hat`	Array of the same dimension as `x.hat` containing the updated cell probabilities.
`error.margins`	List returning, for each margin, the absolute maximum deviation between the desired and generated margin.
`conv`	Boolean indicating whether the algorithm converged to a solution.
`evol.stp.crit`	Vector containing the values of the stopping criterion over the iterations if the selected method is `"ipfp"`.
`solnp.res`	The estimation process uses the `solnp` optimization function from the R package **Rsolnp** and `solnp.res` is the corresponding object returned by the solver (if the selected method is not `"ipfp"`).
`method`	The selected method for estimation.
`call`	The matched call.
If `keep.input = TRUE`:	
`seed`	The original `seed`.
`target.list`	The original `target.list`.
`target.data`	The original `target.data`.

Table 5: Components of the object of class '`mipfp`' returned by the function `Estimate()`.

Name	Description
`object`	An object of class '`mipfp`', usually a result of a call to `Estimate()`.
`cov.method`	Indicates which method to use to compute the covariance. Possible values are Delta (`"delta"`, default) or Lang (`"lang"`).
`prop`	If set to `FALSE` (the default), the results return counts, probabilities otherwise.
`target.list`	The list of the dimensions of the targets used for the estimation process (see `Estimate()` for more details).
`l.names`	If set to a value greater than 0, then the names of the categories will be shortened to a length of `l.names` characters.
`...`	Further optional arguments that can be passed to the underlying `print` and `flat` methods, or to other methods. See their respective documentation for more details (Barthélemy and Suesse 2018).

Table 6: List of arguments for the S3 `summary()` method for '`mipfp`' objects.

Name	Description
`call`	A call object in which all the specified arguments are given by their full names.
`conv`	A Boolean indicating if the specified method converged to a solution (`TRUE`) or not (`FALSE`).
`method`	The method used to generate estimates.
`df`	Degrees of freedom of the estimates.
`estimates`	An array containing the estimates generated by the selected method with their standard deviations and associated t- and p values.
`error.margins`	A list returning, for each margin, the absolute maximum deviation between the desired and generated margin.
`vcov`	A covariance matrix of the estimates (last index moves fastest) computed using the method specified in `cov.method`.
`tab.gof`	A table containing the log-likelihood (`G2`), Wald (`W2`) and Pearson chi-squared (`X2`) statistics with their associated p values.
`stats.df`	Degrees of freedom for the `G2`, `W2` and `X2` statistics.
`dim.names`	Original dimension names of the estimated table.
`l.names`	The value of the parameter `l.names`.

Table 7: Components of the object of class '`summary.mipfp`' returned by the S3 method `summary()` for '`mipfp`' objects.

Name	Description
`object`	The '`mipfp`' object containing the estimates.
`parm`	A specification of which estimates are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all estimates are considered.
`level`	The confidence level required.
`prop`	A Boolean indicating if the results should be using counts (`FALSE`) or proportions (`TRUE`). Default is `FALSE`.
`...`	Further optional arguments passed to other methods (for instance the S3 `vcov` method for '`mipfp`' objects, see Barthélemy and Suesse 2018).

Table 8: List of arguments for the S3 `confint()` method for '`mipfp`' objects.

Name	Description
`n`	Desired sample size. Default = 1.
`mult.bin.dist`	A list describing the multivariate binary distribution. It can be generated by `ObtainMultBinaryDist()`. The list contains at least the element `joint.proba`, an array detailing the joint probabilities of the K binary variables. The array has K dimensions of size 2, referring to the 2 possible outcomes of the considered variable. Hence, the total number of elements is 2^K. Additionally the list can provide the element `var.label`, a list containing the names of the K variables.
`target.values`	A list describing the possibles outcomes of each binary variable, for instance $\{1, 2\}$. Default = $\{0, 1\}$.

Table 9: List of arguments for the function `RMultBinary()`.

Name	Description
`binary.sequences`	The generated $K \times n$ random sequence.
`possible.binary.sequences`	The set of possible binary sequences, i.e., the domain.
`chosen.random.index`	The index of the random draws in the domain.

Table 10: Components of the list returned by the function `RMultBinary()`.

Name	Description
`odds`	A $K \times K$ matrix where the ith row and the jth column represents the odds ratio between variables i and j. Must be provided if `corr` is not.
`corr`	A $K \times K$ matrix where the ith row and the jth column represents the correlation between variables i and j. Must be provided if `odds` is not.
`marg.probs`	A vector with K elements of marginal probabilities where the ith entry refers to $\mathsf{P}(X_i = 1)$.
`...`	Additional arguments that can be passed to the `Ipfp` function such as `tol`, `iter`, `print` and `compute.cov`.

Table 11: List of arguments for the function `ObtainMultBinary()`.

Name	Description
`joint.proba`	The resulting multivariate joint probabilities (from `Ipfp`).
`stp.crit`	The final value of the `Ipfp` stopping criterion.
`check.margins`	A list returning, for each margin, the absolute maximum deviation between the desired and generated margin. Ideally the elements should be close to 0 (from `Ipfp`).

Table 12: Components of the list returned by the function `ObtainMultBinary()`.

Affiliation:

Johan Barthélemy
SMART Infrastructure Facility
University of Wollongong
2522 NSW, Australia
E-mail: johan_barthelemy@uow.edu.au

Thomas Suesse
National Institute for Applied Statistics Research Australia
University of Wollongong
2522 NSW, Australia
E-mail: tsuesse@uow.edu.au

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
August 2018, Volume 86, Code Snippet 2 *Submitted:* 2015-09-27
doi:10.18637/jss.v086.c02 *Accepted:* 2017-09-05