revengc: An R package to reverse engineer summarized data

Samantha Duchscherer, Robert Stewart, and Marie Urban
Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, TN 37831

Abstract

Decoupled (e.g. separate averages) and censored (e.g. > 100 species) variables are continually reported by many well-established organizations (e.g. World Health Organization (WHO), Centers for Disease Control and Prevention (CDC), World Bank, and various national censuses). The challenge therefore is to infer what the original data could have been given summarized information. We present an R package that reverse engineers censored and/or decoupled count data with two main functions. The *cnbinom.pars()* function estimates the average and dispersion parameter of a censored univariate frequency table. The *rec()* function reverse engineers summarized data into an uncensored bivariate table of probabilities.

1 Introduction

The **revengc** R package was originally developed to help model building occupancy [1]. Household size and area of residential structures are typically found in any given national census. If a census revealed the raw data or provided a full uncensored contingency table (household size $*$ area), computing interior density as people per area would be straightforward. However, household size and area are often reported as decoupled variables (separate univariate frequency tables, average values, or a combination of the two). Furthermore, if a contingency table is provided, it typically left ($<$, $\leq$), right ($>$, $\geq$, $+$), and interval (-) censored. This summarized information is problematic for numerous reasons. How can a people per area ratio be calculated when no affiliation between the variables exist? If a census reports a household size average of 5.3, then how many houses are there with 1 person, 2 people,..., 10 people? If a census reports that there are 100 houses in an area of 26-50 square meters, then how many houses are in 26, 27,..., 50 square meters?

A tool that approximates negative binomial parameters from a censored univariate frequency table as well as estimates interior cells of a contingency table governed by negative binomial and/or Poisson marginals can also be useful for other areas ranging from demographic and epidemiological data to ecological inference problems. For example, population and community ecologist could unpack censored organism counts (e.g. < 20 species). Modeling life expectancy and mortality, which are two variables that are notorious for being summarized in an average and/or censored frequency table form, could also benefit from **revengc**. Other summarized examples include the average number of births, the number of new disease cases (censored table), the number of mutations in a gene (censored table) or average mutation rate, etc. We attempt to accommodate for various application of count data by offering five scenarios that can be reverse engineered:

1. *cnbinom.pars()* - An univariate frequency table estimates an average and dispersion parameter
2. *rec()* - Decoupled averages estimates an uncensored contingency table of probabilities
3. *rec()* - Decoupled frequency tables estimates an uncensored contingency table of probabilities
4. *rec()* - An average and frequency table estimates an uncensored contingency table of probabilities
5. *rec()* - A censored contingency table estimates an uncensored contingency table of probabilities

This paper proceeds with our reverse engineering methodology for the two main function: *cnbinom.pars()* and *rec()*. We provide an in-depth analysis of how we implemented both the negative binomial and Poisson distribution as well as the **truncdist** and **mipfp** R package [2, 3]. Since the **revengc** package has specific input requirements, we continue with an explanation of how to implement *cnbinom.pars()* and *rec()* with correctly formatted table(s). We then provide coded examples that implements **revengc** on national census data (household size and area) and end with concluding remarks.

2 The Methodology: *cnbinom.pars()*

The methodology for the *cnbinom.pars()* function is relatively straightforward. To estimate an average μ and dispersion r parameter, a censored frequency table is fit to a negative binomial distribution using a maximum log-likelihood function customized to handle left ($<$, $\leq$), right ($>$, $\geq$, $+$), and interval (-) censored data. To show an example, first recall the negative binomial distribution $P(X = x|\mu, r)$ parameterized as a distribution of the number of failures X before the r^{th} success in independent trials (1). With success probability p in each trail, $r \geq 0$ and $0 \leq p \leq 1$ [4].

$$\begin{aligned} P(X = x|r, p) = \binom{x + r - 1}{x} p^r (1-p)^y \equiv P(X = x|r, \mu) \binom{x + r - 1}{x} \left(\frac{r}{\mu + r}\right)^r \left(\frac{\mu}{\mu + r}\right)^y \\ E(X) = \frac{r(1-p)}{p} = \mu \\ V(X) = \frac{r(1-p)}{p^2} = \mu + \frac{\mu^2}{r} \end{aligned} \tag{1}$$

Now consider an arbitrary censored frequency table x that has a combination of left censored ($x < c$), interval censored ($a \leq x \leq b$), and right censored ($x > d$) data (i.e. *a, b, c,* and *d* represent the censoring limits). The optimal μ and r parameter for x maximizes its custom log-likelihood function (2).

$$\begin{aligned} L(\mu, r|x)_{\log} = \\ &+ \sum_{x<c} \log\left(P\left(x < c|\mu, r\right)\right) \\ &+ \sum_{a \leq x \leq b} \log\left(P\left(a \leq x \leq b|\mu, r\right)\right) \\ &+ \sum_{x>d} \log\left(P\left(x > d|\mu, r\right)\right) \end{aligned} \tag{2}$$

3 The Methodology: *rec()*

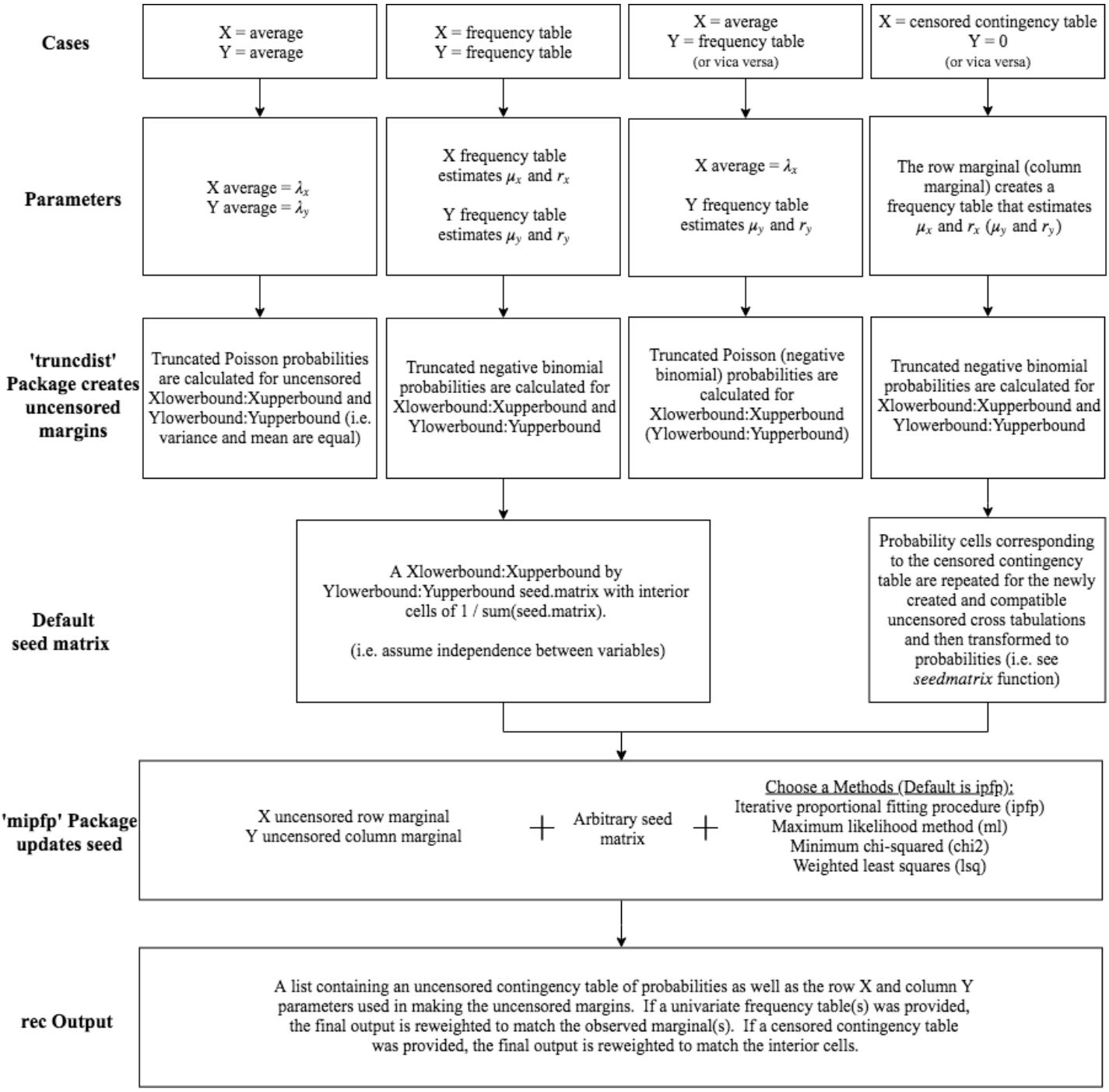
3.1 Overview

rec() is a statistical approach that estimates the probabilities of a 'true' contingency table given summarized information: two averages, two univariate frequency tables, a combination of an average and univariate frequency table, and a censored contingency table. Figure 1 presents a methodology workflow.

3.2 Negative Binomial and Poisson Distribution

When only an average is provided, we assume the average and variance are equal and rely on a Poisson distribution (i.e. the probability of observing x events in a given interval is given by Equation 3). We understand that there are many cases where data has more variation than what is indicated by the Poisson distribution (i.e. overdispersion). However, with limited data, the Poisson distribution is implemented due to its convenient property of having only one parameter, λ = average. For the cases with more data (i.e. univariate frequency table(s) or censored contingency table), we account for dispersion by relying on the more flexible negative binomial distribution (1). Hence, in these cases, the *cnbinom.pars()* function estimates the optimal average μ and dispersion r parameters.

$$\begin{aligned} P\left(X = x\right) = e^{-\lambda} \frac{\lambda^x}{x!} \\ E(X) = \lambda \\ V(X) = \lambda \end{aligned} \tag{3}$$



Workflow of rec() function.

3.3 truncdist R package

With the negative binomial (μ and r) and/or Poisson (λ) parameters, *rec()* calculates truncated distributions to represent uncensored row (Xlowerbound:Xupperbound) and column (Ylowerbound:Yupperbound) margins. Calculations use the **truncdist** R package [2], and to provide a reference, Equation 4 gives the probability density function of a truncated X distribution over the interval *(a,b]* (i.e. the negative binomial and/or Poisson probability density function is represented by $g(\cdot)$ and their corresponding cumulative distribution function is denoted by $G(\cdot)$). Note, truncated distributions are very practical in this context because the distributions (margins) are restricted to a desired row and column marginal length.

$$f_X(x) = \begin{cases} \frac{g(x)}{G(b)-G(a)}, & \text{if } a < x \leq b \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The *(a,b]* interval needed for both X (row of contingency table) and Y (column of contingency table) can be selected intuitively or with a brute force method. If *rec()* outputs a final contingency table with higher probabilities near the edge(s) of the table, then it would make sense to increase the range of the bound(s). For both variables, this would just involve making the lower bound less, making the upper bound more, or doing a combination of the two. The opposite holds true as well. If the final contingency table in *rec()* has very low probabilities near the edge(s) of the table, the range of the particular bound(s) should be decreased.

3.4 mipfp R package

rec() utilizing the **mipfp** R package to create cross tabulation probability estimates. As mentioned above, the row and column marginals are the uncensored truncated distributions. However, an opportunity for sensitivity analysis is presented by allowing an arbitrary seed estimation method and seed matrix. Focusing on the seed estimation methods first, **mipfp** provides four algorithms (Table 1). Essentially, all methods adjust cell proportions p_{xy} in a X $*$ Y contingency table to known marginal probabilities π_{x+} and π_{+y} (i.e. all interior cell estimates $\hat{\pi}_{xy}$ are subject to marginal constraints (5)). For a better understanding, please refer the papers by [5] and [6].

$$\begin{aligned} \sum_y \hat{\pi}_{x+} \qquad & (x = 1, ..., X) \\ \sum_x \hat{\pi}_{+y} \qquad & (y = 1, ..., Y) \end{aligned} \tag{5}$$

Method	**Abbreviation**	**Calculate $\hat{\pi}_{xy}$ by**
Iterative proportional fitting procedure	ipfp	Minimizing $\sum_x \sum_y \hat{\pi}_{xy} ln(\hat{\pi}_{xy}/p_{xy})$
Maximum likelihood method	ml	Maximizing $\sum_x \sum_y p_{xy} ln(\hat{\pi}_{xy})$
Minimum chi-squared	chi2	Minimizing $\sum_x \sum_j (\hat{\pi}_{xy} - p_{xy})^2/\hat{\pi}_{xy}$
Weighted least squares	lsq	Minimizing $\sum_x \sum_y (p_{xy} - \hat{\pi}_{xy})^2/p_{xy}$

mipfp methods to generate estimated cross tabulations [3].

Now considering the arbitrary seed matrix, *rec()* provides reasonable defaults. For the decoupled cases (two averages, two tables, or a combination of a table and average), the absence of additional information makes it difficult to say much about the joint distribution. Therefore, *rec()* assumes independence between

the variables, which is equivalent in making the X $*$ Y seed a matrix of ones (i.e. probability of this seed.matrix is 1 / sum(seed.matrix)). When a censored contingency table is provided, independence does not have to be assumed and the interior cells can be weighted. *rec()* creates the default seed matrix by first repeating probability cells, which corresponding to the censored contingency table, for the newly created and compatible uncensored cross tabulations. Each cell value j of this new seed.matrix is then changed to a probability $seed.matrix[j]/sum(seed.matrix)$. To see the seed for this case, refer to the Example section (Indonesia).

4 Usage

4.1 cnbinom.pars()

The *cnbinom.pars()* function has the following format with a description of the argument directly below. The output is a list consisting of an estimated average (mu) and dispersion (r) parameter.

```
cnbinom.pars(censoredtable)
```

censoredtable: A frequency table (censored and/or uncensored). A data.frame and matrix are acceptable classes. See **Data entry** section for formatting.

4.2 rec()

The *rec()* function has the following format with a description of each argument directly below. The output is a list containing an uncensored contingency table of probabilities (rows range from Xlowerbound:Xupperbound and the columns range from Ylowerbound:Yupperbound) as well as the row and column parameters used in making the margins for the **mipfp** R package.

```
rec(X, Y, Xlowerbound, Xupperbound, Ylowerbound, Yupperbound,
  seed.matrix, seed.estimation.method)
```

X: Argument can be an average, a univariate frequency table, or a censored contingency table. The average value should be a numeric class while a data.frame or matrix are acceptable table classes. Y defaults to NULL if X argument is a censored contingency table. See **Data entry** section for formatting.

Y: Same description as X but this argument is for the Y variable. X defaults to NULL if Y argument is a censored contingency table.

Xlowerbound: A numeric class value to represent the left bound for X (row in contingency table). The value must strictly be a non-negative integer and cannot be greater than the lowest category/average value provided for X (e.g. the lower bound cannot be 6 if a table has '≥ 5' as a X or row category).

Xupperbound: A numeric class value to represent the right bound for X (row in contingency table). The value must strictly be a non-negative integer and cannot be less than the highest category/average value provided for X (e.g. the upper bound cannot be 90 if a table has '> 100' as a X or row category).

Ylowerbound: Same description as Xlowerbound but this argument is for Y (column in contingency table).

Yupperbound: Same description as Xupperbound but this argument is for Y (column in contingency table).

seed.matrix: An intial probability matrix to be updated. If decoupled variables is provided the default is a

Xlowerbound:Xupperbound by Ylowerbound:Yupperbound seed.matrix with interior cells of 1 / sum(seed.matrix). If a censored contingency table is provided the default is the $seedmatrix()\$Probabilities$ output.

seed.estimation.method: A character string indicating which method is used for updating the seed.matrix. The choices are: "ipfp", "ml", "chi2", or "lsq". Default is "ipfp".

5 Data entry

The input tables are formatted to accommodate most open source data. The univariate frequency table used in *cnbinom.pars()* and/or *rec()* needs to be a data.frame or matrix class with two columns and n rows. The categories must be in the first column with the frequencies or probabilities in the second column. Row names should never be placed in this table (the default row names should always be 1:n). Column names can be any character string. The only symbols accepted for censored data are listed below. Note, less than or equal to ($\leq$ and LE) is not equivalent to less than ($<$ and L) and greater than or equal to ($\geq$, +, and GE) is not equivalent to greater than ($>$ and G). Also, **revengc** uses closed intervals.

- **Left censored symbols:** $<$, $\leq$, L, and LE
- **Interval censored symbols:** $-$ and I (symbol has to be placed in the middle of the two category values)
- **Right censored symbols:** $>$, $\geq$, +, G, and GE
- **Uncensored symbol:** no symbol (only provide category value)

To provide examples, the three tables below use different censored symbols yet give the same *cnbinom.pars()* output 2.

Category	Frequency
≤ 6	11800
7-12	57100
13-19	14800
20+	3900

Category	Frequency
LE 6	11800
7 I 12	57100
13 I 19	14800
GE 20	3900

Category	Frequency
< 7	11800
7 I 12	57100
13-19	14800
≥ 20	3900

Examples of correctly formatted univariate tables.

The censored contingency table for *rec()* has a similar format. The censored symbols should follow the requirements listed above. The table's class can be a data.frame or a matrix. The column names should be the Y category values. The first column should be the X category values and the row names can be arbitrary. The inside of the table are X $*$ Y frequencies, which are either non-negative frequencies or probabilities if seed.estimation.method is "ipfp" or strictly positive when method is "ml", "lsq" or "chi2". The row and column marginal totals corresponding to their X and Y category values need to be placed in this table. The top left, top right, and bottom left corners of the table should be NA or blank. The bottom right corner can be a total cross tabulation sum value, NA, or blank. Table 3 is a formatted example.

5.1 Formatting tables in R

The code below shows how to format these tables properly in R.

NA	<20	20-30	>30	NA
<5	18	19	8	45
5-9	13	8	12	33
≥10	7	5	10	21
NA	38	32	31	NA

Example of a correctly formatted bivariate table.

```
# read in csv file
# univariatetable.csv is a preloaded example
# univariatetable.csv<-read.csv("filename.csv", row.names = NULL,
#  header= FALSE, check.names=FALSE)

# create univariate table
univariatetable<-cbind(as.character(c("1-2", "3-4", "5-6", "7-8", ">=9")),
  c(16.2, 41.7, 29.0, 9.0, 4.1))

# create contingency table
# contingencytable.csv is a preloaded example that provides the same table
contingencytable<-matrix(c(6185,9797,16809,11126,6156,3637,908,147,69,4,
  5408,12748,26506,21486,14018,9165,2658,567,196,78,
  7403,20444,44370,36285,23576,15750,4715,994,364,136,
  4793,17376,44065,40751,28900,20404,6557,1296,555,228,
  2354,11143,32837,33910,26203,19301,6835,1438,618,245,
  1060,6038,19256,21298,17774,13864,4656,1039,430,178,
  273,2521,9110,11188,9626,7433,2608,578,196,112,
  119,1130,4183,5566,5053,3938,1367,318,119,66,
  33,388,1707,2367,2328,1972,719,171,68,37,
  38,178,1047,1672,1740,1666,757,193,158,164),
  nrow=10,ncol=10, byrow=TRUE)
rowmarginal<-apply(contingencytable,1,sum)
contingencytable<-cbind(contingencytable, rowmarginal)
colmarginal<-apply(contingencytable,2,sum)
contingencytable<-rbind(contingencytable, colmarginal)
row.names(contingencytable)[row.names(contingencytable)=="colmarginal"]<-""
contingencytable<-data.frame(c("1","2","3","4","5","6", "7", "8","9","10+", NA),
  contingencytable)
colnames(contingencytable)<-c(NA,"<20","20-29","30-39","40-49","50-69","70-99",
  "100-149","150-199","200-299","300+", NA)
```

6 Worked examples

6.1 Nepal

A Nepal Living Standards Survey [7] provides both a censored table and average for urban household size. We use the censored table to show that the *cnbinom.pars()* function calculates a close approximation to the provided average household size (4.4 people). Note, there is overdispersion in the data.

```
# revengc has the Nepal household table preloaded as univariatetable.csv
cnbinom.pars(censoredtable = univariatetable.csv)
```

6.2 Indonesia

In 2010, the Population Census Data - Statistics Indonesia provided over 60 censored contingency tables containing Floor Area of Dwelling Unit (square meter) by Household Member Size. The tables are separated by province, urban, and rural. Here we use the household size by area contingency table for Indonesia's rural Aceh Province to show the multiple coding steps and functions implemented inside *rec()*. This allows the user to see a methodology workflow in code form. The final uncensored household size by area estimated probability table, which implemented the 'ipfp' seed estimation method and default seed matrix, has rows ranging from 1 (Xlowerbound) to 15 (Xupperbound) people and columns ranging from 10 (Ylowerbound) to 310 (Yupperbound) square meters.

```
# data = Indonesia 's rural Aceh Province censored contingency table
# preloaded as 'contingencytable.csv'
contingencytable.csv

# provided upper and lower bound values for table
# X=row and Y=column
Xlowerbound=1
Xupperbound=15
Ylowerbound=10
Yupperbound=310

# table of row marginals provides average and dispersion for x
row.marginal.table<-row.marginal(contingencytable.csv)
x<-cnbinom.pars(row.marginal.table)
# table of column marginals provides average and dispersion for y
column.marginal.table<-column.marginal(contingencytable.csv)
y<-cnbinom.pars(column.marginal.table)

# create uncensored row and column ranges
rowrange<-Xlowerbound:Xupperbound
colrange<-Ylowerbound:Yupperbound

# new uncensored row marginal table = truncated negative binomial distribution
uncensored.row.margin<-dtrunc(rowrange, mu=x$Average, size = x$Dispersion,
  a = Xlowerbound-1, b = Xupperbound, spec = "nbinom")
# new uncensored column margin table = truncated negative binomial distribution
uncensored.column.margin<-dtrunc(colrange, mu=y$Average, size = y$Dispersion,
  a = Ylowerbound-1, b = Yupperbound, spec = "nbinom")

# sum of truncated distributions equal 1
# margins need to be equal for mipfp
sum(uncensored.row.margin)
sum(uncensored.column.margin)

# create seed of probabilities (rec() default)
seed.output<-seedmatrix(contingencytable.csv, Xlowerbound,
  Xupperbound, Ylowerbound, Yupperbound)$Probabilities

# run mipfp
# store the new margins in a list
tgt.data<-list(uncensored.row.margin, uncensored.column.margin)
# list of dimensions of each marginal constrain
tgt.list<-list(1,2)
# calling the estimated function
```

```
## seed has to be in array format for mipfp package
## ipfp is the selected seed.estimation.method
final1<-Estimate(array(seed.output,dim=c(length(Xlowerbound:Xupperbound),
  length(Ylowerbound:Yupperbound))), tgt.list, tgt.data, method="ipfp")$x.hat

# filling in names of updated seed
final1<-data.frame(final1)
row.names(final1)<-Xlowerbound:Xupperbound
names(final1)<-Ylowerbound:Yupperbound

# reweight estimates to known censored interior cells
final1<-reweight.contingencytable(observed.table = contingencytable.csv,
  estimated.table = final1)

# final results are probabilities
sum(final1)

# rec() function outputs the same table
# default of rec() seed.estimation.method is ipfp
# default of rec() seed.matrix is the output of seedmatrix()$Probabilities
final2<-rec(X= contingencytable.csv,
  Xlowerbound = 1,
  Xupperbound = 15,
  Ylowerbound = 10,
  Yupperbound = 310)

# check that both data.frame results have same values
all(final1 == final2$Probability.Estimates)
```

7 Conclusion

revengc was designed to reverse engineer summarized and decoupled variables with two main functions: *cnbinom.pars()* and *rec()*. Relying on a negative binomial distribution, *cnbinom.pars()* approximates the average and dispersion parameter of a censored univariate frequency table. *rec()* fills in missing interior cell values from observed aggregated data (i.e. decouples average(s) and/or censored frequency table(s) or a censored contingency table). There are some required assumptions in *rec()*. For instance, *rec()* relies on a Poisson distribution when only an average is provided, which is assuming the variance and average are equal. More descriptive input variables, such as univariate tables or contingency tables, can account for dispersion found in data. However, independence between decoupled variables still has to be assumed when there is no external information about the joint distribution. For these reasons, **rec()** provides two options for sensitivity analysis: the seed matrix and the method used in updating the seed matrix are both arbitrary inputs.

References

[1] R. Stewart, M. Urban, S. Duchscherer, J. Kaufman, A. Morton, G. Thakur, J. Piburn, and J. Moehl, "A bayesian machine learning model for estimating building occupancy from open source data," *Natural Hazards*, vol. 81, no. 3, pp. 1929–1956, 2016.

[2] F. Novomestky and S. Nadarajah, *truncdist: Truncated Random Variables*, 2016. R package version 1.0-2.

[3] Johan Barthélemy, Thomas Suesse, Mohammad Namazi-Rad, *Multidimensional Iterative Proportional Fitting and Alternative Models*. R Foundation for Statistical Computing, 2018.

[4] A. Lindén and S. Mäntyniemi, "Using the negative binomial distribution to model overdispersion in ecological count data," *Ecology*, vol. 92, no. 7, pp. 1414–1421, 2011.

[5] R. J. Little and M.-M. Wu, "Models for contingency tables with known margins when target and sampled populations differ," *Journal of the American Statistical Association*, vol. 86, no. 413, pp. 87–95, 1991.

[6] T. Suesse, M.-R. Namazi-Rad, P. Mokhtarian, and J. Barthélemy, "Estimating cross-classified population counts of multidimensional tables: an application to regional australia to obtain pseudo-census counts," vol. 33, no. 4, 2017.

[7] N. P. C. S. Government of Nepal, "Nepal living standards survey," tech. rep., Central Bureau of Statistics, 11 2011.