Gentile, G, et al. 2020 CURSAT ver. 2.1: A Simple, Resampling-Based, Program to Generate Pseudoreplicates of Data and Calculate Rarefaction Curves. *Journal of Open Research Software,* 8: 17. DOI: https://doi.org/10.5334/jors.260

SOFTWARE METAPAPER

CURSAT ver. 2.1: A Simple, Resampling-Based, Program to Generate Pseudoreplicates of Data and Calculate Rarefaction Curves

Gabriele Gentile
Department of Biology, University of Rome Tor Vergata, Rome, IT
gabriele.gentile@uniroma2.it

CURSAT ver. 2.1 is an open-source code in QB64 basic, compilable into an executable file, that produces n pseudoreplicates of an empirical data set. Both resamplings with and without replacement are allowed by the software. The number (n) of pseudoreplicates is set by the user. Pseudoreplicates can be exported in a file that can be opened by a spreadsheet. Thus, pseudoreplicates are permanently stored and available for the calculation of statistics of interest and associated variance. The software also uses the n pseudoreplicate data to reconstruct n accumulation matrices, appended in an output file. Accumulation has applicability in cases in which repeated sample-based data must be evaluated for exhaustiveness. Many situations involve repeated sampling from the same set of observations. For example, if data consist of species occurrence, the software can be used by a wide spectrum of specialists such as ecologists, zoologists, botanists, biogeographers, conservationists for biodiversity estimation. The software allows performing accumulation irrespectively whether the input data set contains abundance (quantitative) or incidence (binary) data. Accumulation matrices can be imported in statistical packages to estimate distributions of successive pooling of samples and depict accumulation and rarefaction curves with associated variance.

CURSAT ver. 2.1 is released in two editions. Edition #1 is recommended for analysis, whereas Edition #2 generates a log file in which the flow of internal steps of resampling and accumulation routines is reported. Edition #2 is primarily designed for educational purposes and quality check.

Keywords: Saturation; accumulation; pseudoreplicates; rarefaction curves; repeated sampling; replacement; non-random sampling; bootstrap; ecology; biogeography; species-area; zoology; botany; biodiversity; diversity indices
Funding statement: The software was developed with no specific funds.

(1) Overview

Introduction

Many situations involve repeated sampling from the same set of observations. For example, Ridenhour and Grimmett [1] hypothesized the case of a professor who "*has a test bank of 100 questions for a particular course and randomly chooses 25 of these questions for the final exam each semester. A persistent but not very talented student repeats the course several times. Obviously, the student has no chance of having seen all the questions before taking the course four times. What is the probability that the student will have seen all the questions after k repetitions? That is, what is the probability that the entire test bank will have been exhausted after k repetitions?*". Clearly, we could also reformulate the question as "how many *k* samples of size *n* are necessary for the test bank to be exhausted?". This implies selecting *k* samples of size *n* from a population containing *N* members. In similar cases, when *N* and *n* are known, the issue can be tackled by using a probabilistic approach [1].

When *N* and *n* are unknown accumulation procedures allow the construction of rarefaction (saturation) curves that may help to evaluate the exhaustiveness of sampling.

In ecology, accumulation procedures associated with resampling methods may offer support for measuring biodiversity. In particular, assessing the number of species that occur in a certain area (species richness) is a central issue in ecology because such information participates in defining a number of diversity estimators used in describing and comparing profiles of biodiversity [2]. Estimators can be based on the abundance of individuals belonging to a certain class (species) in a sample [3, 4]. Additionally, they can also be based on incidence, which is the presence-absence data of a class (species) in a given sample [5].

Independently whether it is based on abundance or incidence, the estimation of species richness depends on sample size, due to both sampling effects and intrinsic factors such as seasonality, species turnover, etc. As a consequence of this, repeated sampling is expected to increase the cumulative number of species observed in a certain area. Thus, the successive pooling of samples from a single location produces a species accumulation, and a pattern of the species accumulation and rarefaction curves may be described in several model-based ways [6, 7, 8, 9].

An alternative to model-based ways to estimate accumulation and rarefaction curves is provided by resampling. Resampling with or without replacement has become one of the most widely used measures of statistical support in several disciplines, especially when the analytical estimation of the variance of statistics of interest is not possible. The second option is usually preferred over the first, because resampling with replacement (bootstrap) may suffer some underestimation respect to resampling without replacement. However, the variance among randomizations estimated via resampling without replacement approaches zero, when reaching the last accumulation level, whereas variance estimated via bootstrap does not suffer from such an effect [10].

Resampling produces what is defined as a pseudoreplicate data set in which columns of the original data matrix may be represented multiple times, not at all, or simply in a different order. Each pseudoreplicate can be analyzed using the same statistics as for the original data matrix. Reiteration for a number of times creates a distribution of the statistics investigated. Such distribution may serve as a basis for statistical testing.

Three major schemes of resampling are known: bootstrap, jackknife, and shuffling. Bootstrapping implies resampling with replacement, whereas jackknifing and shuffling do not. Bootstrapping produces pseudoreplicate data sets in which columns of the original data matrix may be represented multiple times or not at all. By jackknifing, a proportion of columns in the original data set are deleted and not replaced whereas the shuffling procedure simply changes the order by which columns are represented in the pseudoreplicate data set, without deleting any columns. Bootstrapping and jackknifing tend to produce similar results [11].

Resampling methods assume that elements in a data set must be identically and independently distributed. Such an assumption may be unrealistic, although random resampling would allow such conditions to be met [11].

CURSAT ver. 2.1 is a simple open-source software that allows the generation of exportable pseudoreplicate data sets and, in combination with commonly used statistical packages, the construction of accumulation and rarefaction curves and associated sample variance by using a rarefaction-by-resampling approach. Such an approach is not novel and has already been applied in ecology and genetics (e.g., [12, 13]). In fact, several R packages and software implement resampling rarefaction approaches such as the specaccum function in vegan R package [14], mothur [15], the EstimateS program [16], iNEXT R package [17], and RTK [18].

Whereas CURSAT ver.2.1 is not meant to compete with other more comprehensive and mature software available, especially when dealing with extremely large data sets, it enables to extract replicated data matrices, while other packages do not. Admittedly, for programmers, R requires minor programming efforts to export replicated data matrices resulting from resampling routines. However, CURSAT ver. 2.1 may still prove handy because it does not require previous experience in R or other programming software. It can be run as an executable file and allows the user to save both pseudoreplicate and accumulation data in separate files. This is a valuable feature because pseudoreplicate data may be imported into a spreadsheet and be subsequently used for the calculation of diversity indices or other statistics with associated resampling-based variance in case analytical variance can not be estimated. By exporting pseudoreplicate data sets, the user has the possibility to estimate different statistics using the same pseudoreplicate data set. Additionally, runs are reproducible because the user can set a seed number.

To promote a full understanding of procedures followed by the software during calculations, extended remarks (REM lines) are provided in the source code. The REM lines in the source code explain and describe what is performed by subsequent lines of instructions. Additionally, a second edition (Edition #2) of the software is provided. Edition #2 generates a log file where intermediate matrices created and used by the software during resampling and accumulation routines are appended and saved at each replicate. This may prove useful for check-quality and educational purposes.

Implementation and architecture

CURSAT ver. 2.1 is a short open-source code written in basic (QB64), compilable into an executable file. It uses a random sampling approach and allows the user to choose between two options: 1) resampling with replacements (bootstrap) and 2) resampling without replacement (shuffling).

The software produces n resampled replicates of $n \times n$ objects-by-sample matrix. The matrix may incorporate quantitative or qualitative data. In fact, cells may contain counts of individuals per object (abundance) or presence/absence (binary) data. This software is primarily designed to address questions such as: "how many sampling events are necessary to maximize the chance to sample all objects?" or "how many sampling events are necessary to collect a representative sample of objects?".

For example, let's consider the following case, very common in ecological studies. We know that 12 species (objects) occur at a certain location. We sampled that location 10 times and each time we found a certain number of species, according to the following scheme: in the 12 × 10 matrix, species are a-l (rows) and sampling events are 1–10 (columns). The presence and absence (incidence) of each species in each sampling event is represented by 1 and 0, respectively as in **Table 1**.

Table 1: Incidence matrix.

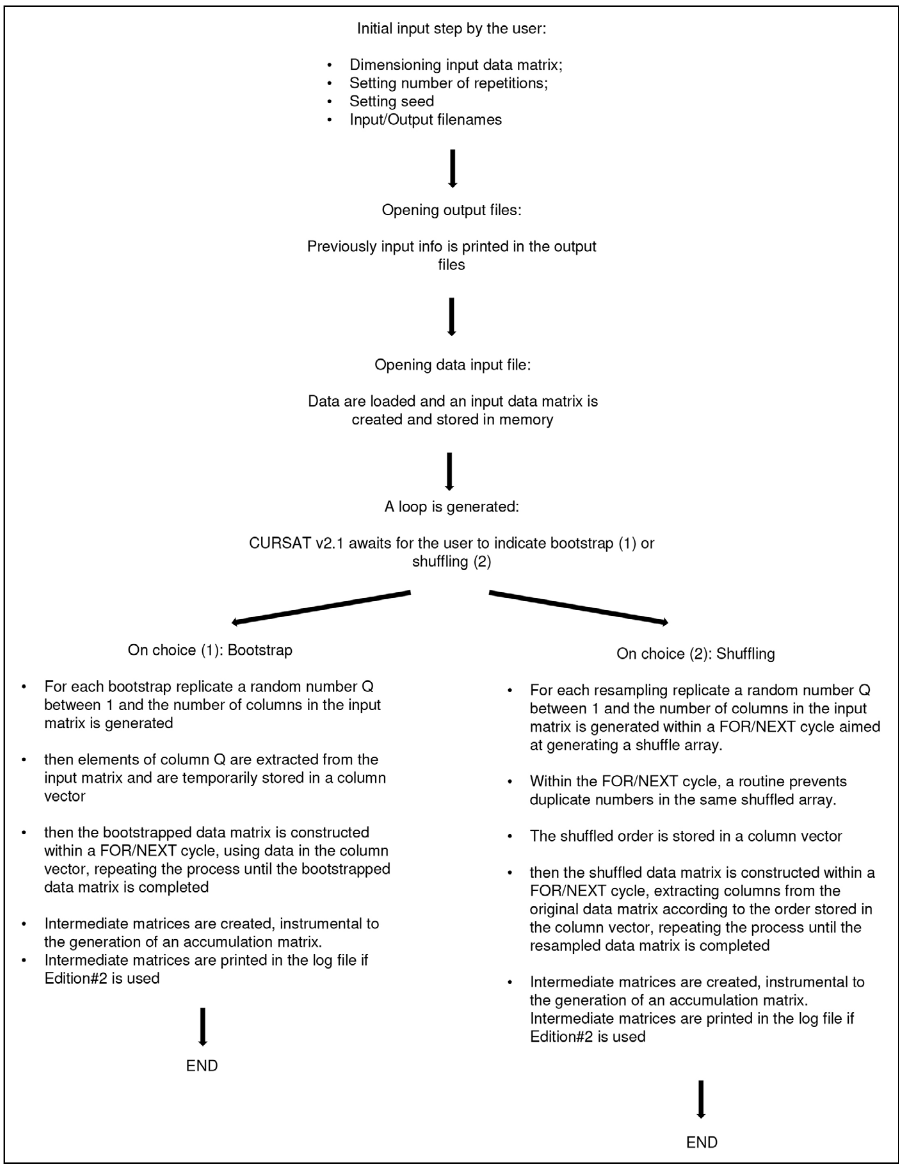
	1	2	3	4	5	6	7	8	9	10
a	1	0	1	1	0	0	0	1	1	0
b	1	1	0	0	0	0	1	1	0	0
c	0	0	0	1	0	1	1	0	1	1
d	0	0	0	0	1	0	1	0	1	0
e	0	0	1	1	0	0	0	1	0	0
f	1	1	0	0	1	0	0	1	0	1
g	0	0	1	1	1	0	0	0	1	0
h	1	1	0	1	0	0	1	1	0	1
i	1	0	0	0	1	1	0	1	0	1
j	0	0	1	1	0	0	1	0	1	0
k	0	0	1	1	0	1	0	1	0	1
l	1	1	1	0	0	0	1	1	0	0

The data matrix is the input information for CURSAT ver. 2.1. Because of the limitation of Basic language, CURSAT ver. 2.1 accepts blanks or commas as separators in the input file. An incidence matrix is used in the example mentioned above, but an abundance matrix may also be used indifferently because the software will handle it when constructing accumulation matrices.

The functional structure of the software is summarized in the flow chart in **Figure 1**.

Resampling with replacement (Bootstrap)

After loading the rectangular input data matrix D, previously dimensioned by the user, CURSAT ver. 2.1 starts the bootstrap by first randomly choosing a number corresponding to one of the columns of D. Subsequently, CURSAT ver. 2.1 extracts from D the elements of that column and starts constructing the bootstrapped data matrix ND using data in D. ND and D matrices have the same dimension. The process is repeated for the number of replicates set by the user.



Initial input step by the user:

- Dimensioning input data matrix;
- Setting number of repetitions;
- Setting seed
- Input/Output filenames

Opening output files:

Previously input info is printed in the output files

Opening data input file:

Data are loaded and an input data matrix is created and stored in memory

A loop is generated:

CURSAT v2.1 awaits for the user to indicate bootstrap (1) or shuffling (2)

On choice (1): Bootstrap

- For each bootstrap replicate a random number Q between 1 and the number of columns in the input matrix is generated
- then elements of column Q are extracted from the input matrix and are temporarily stored in a column vector
- then the bootstrapped data matrix is constructed within a FOR/NEXT cycle, using data in the column vector, repeating the process until the bootstrapped data matrix is completed
- Intermediate matrices are created, instrumental to the generation of an accumulation matrix.
- Intermediate matrices are printed in the log file if Edition#2 is used

END

On choice (2): Shuffling

- For each resampling replicate a random number Q between 1 and the number of columns in the input matrix is generated within a FOR/NEXT cycle aimed at generating a shuffle array.
- Within the FOR/NEXT cycle, a routine prevents duplicate numbers in the same shuffled array.
- The shuffled order is stored in a column vector
- then the shuffled data matrix is constructed within a FOR/NEXT cycle, extracting columns from the original data matrix according to the order stored in the column vector, repeating the process until the resampled data matrix is completed
- Intermediate matrices are created, instrumental to the generation of an accumulation matrix. Intermediate matrices are printed in the log file if Edition#2 is used

END

Figure 1: Functional structure of CURSAT ver.2.1.

Resampling without replacement (Shuffling)
Similarly to the bootstrap routine, after loading the rectangular input data matrix D, CURSAT ver. 2.1 generates a random number comprised between 1 and the maximum number of columns of D, to construct a row vector F with as many elements as the number of columns of D. Such a process occurs within a cycle that accepts only non-replicated numbers. The vector F stores the shuffling order. The remaining part of the routine is similar to the bootstrap routine. CURSAT ver. 2.1 extracts the elements of each column from D, following the order stored in F, and starts constructing the resampled data matrix ND using data in D. ND and D matrices have the same dimension. Again, the process is repeated as many times as the number of replicates set by the user.

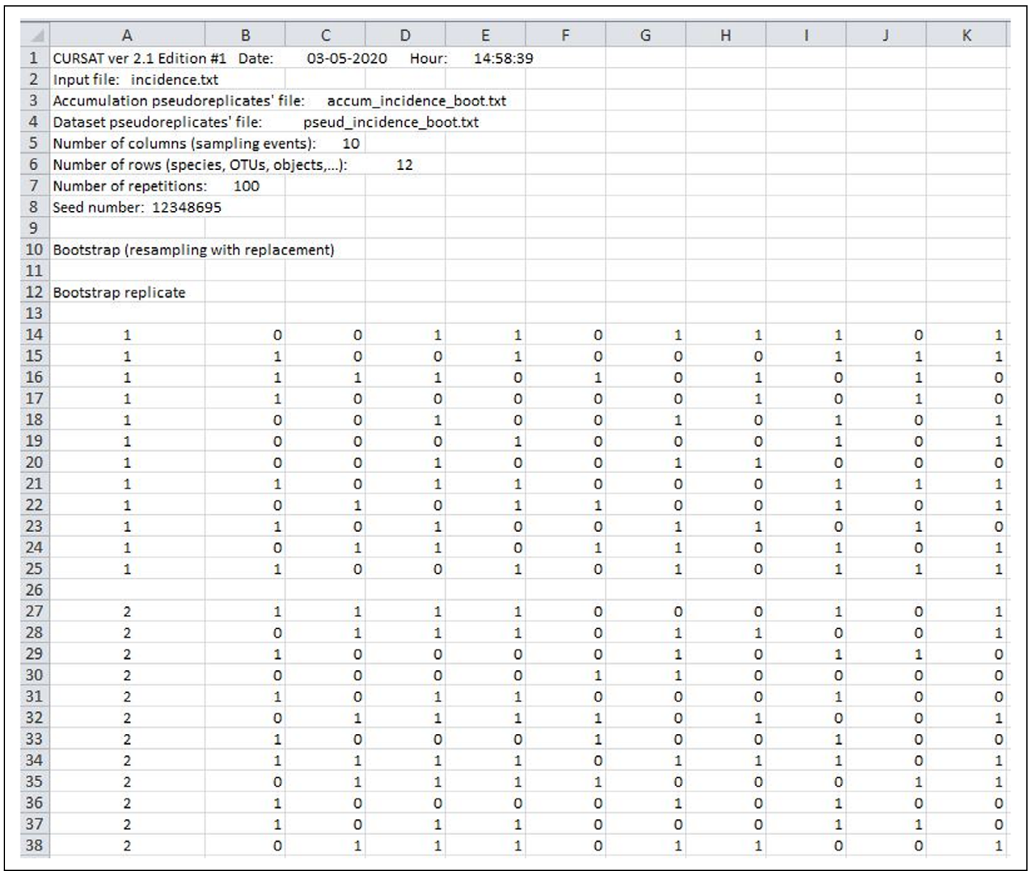
Accumulation
To perform accumulation, for each resampling replicate, CURSAT ver. 2.1 constructs the matrix W where, for each row, each *i*th element is the cumulative sum of the first to the *i*th element of the corresponding row in ND. Matrix W is converted into the binary matrix B by replacing each element larger than 0 with 1. The matrix B is not an incidence matrix *sensu stricto*. In fact, matrix B does not respond to the question: "Is object *x* present in each sampling event?", but rather it answers the question: "Is object *x* present in the cumulative sample after *n* sampling events?". In the last step, for each column of B, elements are cumulatively summed and the accumulation matrix is generated. The stratagem to use matrices W and B is instrumental to correctly perform accumulation, irrespectively whether the input file (matrix D) contains abundance or incidence data.

Number of pseudoreplicates and repeatability
The number (*n*) of pseudoreplicates is set by the user. A scalar seed is also set by the user, to allow repeatability.

The software appends the *n* replicated data matrices in an output file (named by the user) as in **Figure 2**.

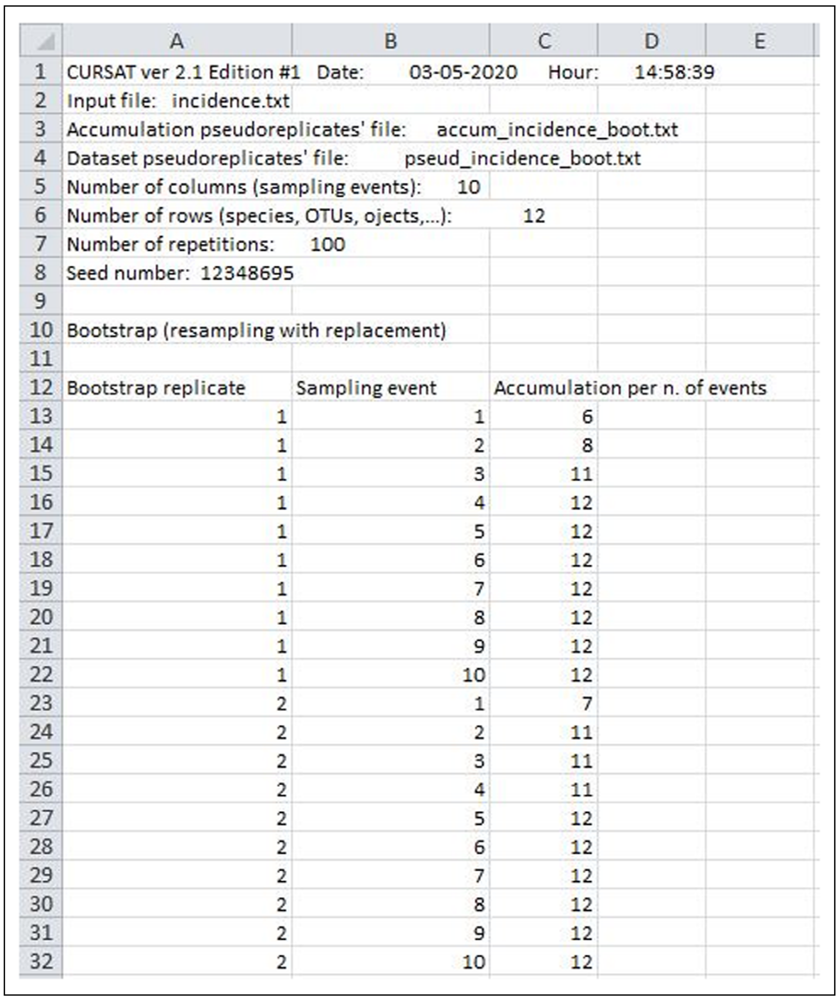
Correspondent *n* sample-based object accumulation matrices are also generated and appended in a file named by the user, as in **Figure 3**. For an easy import of both pseudoreplicate data and accumulation matrices into a spreadsheet, tabulations are used as separators in the output files. Both pseudoreplicate data and accumulation files provide a full report of the run. The first column of each pseudoreplicate data matrices is a label that marks the resample replicate during which the pseudoreplicate was generated.

Similar information is reported in the first column of each of the accumulation matrices generated. In these matrices sampling events and associated cumulative number of objects (species in the example) are reported in the second and third columns, respectively. Such a file can



	A	B	C	D	E	F	G	H	I	J	K
1	CURSAT ver 2.1 Edition #1 Date: 03-05-2020 Hour: 14:58:39										
2	Input file: incidence.txt										
3	Accumulation pseudoreplicates' file: accum_incidence_boot.txt										
4	Dataset pseudoreplicates' file: pseud_incidence_boot.txt										
5	Number of columns (sampling events): 10										
6	Number of rows (species, OTUs, objects,...): 12										
7	Number of repetitions: 100										
8	Seed number: 12348695										
9											
10	Bootstrap (resampling with replacement)										
11											
12	Bootstrap replicate										
13											
14	1	0	0	1	1	0	1	1	1	0	1
15	1	1	0	0	1	0	0	0	1	1	1
16	1	1	1	1	0	1	0	1	0	1	0
17	1	1	0	0	0	0	0	1	0	1	0
18	1	0	0	1	0	0	1	0	1	0	1
19	1	0	0	0	1	0	0	0	1	0	1
20	1	0	0	1	0	0	1	1	0	0	0
21	1	1	0	1	1	0	0	0	1	1	1
22	1	0	1	0	1	1	0	0	1	0	1
23	1	1	0	1	0	0	1	1	0	1	0
24	1	0	1	1	0	1	1	0	1	0	1
25	1	1	0	0	1	0	1	0	1	1	1
26											
27	2	1	1	1	1	0	0	0	1	0	1
28	2	0	1	1	1	0	1	1	0	0	1
29	2	1	0	0	0	0	1	0	1	1	0
30	2	0	0	0	0	1	1	0	0	0	0
31	2	1	0	1	1	0	0	0	1	0	0
32	2	0	1	1	1	1	0	1	0	0	1
33	2	1	0	0	0	1	0	0	1	0	0
34	2	1	1	1	1	0	1	1	1	0	1
35	2	0	1	1	1	1	0	0	0	1	1
36	2	1	0	0	0	0	1	0	1	0	0
37	2	1	0	1	1	0	0	0	1	1	0
38	2	0	1	1	1	0	1	1	0	0	1

Figure 2: File *pseud_incidence_boot.txt* opened in Microsoft Excel. It consists of the output file with 100 bootstrapped pseudoreplicates of the original dataset in the file *incidence.txt*. The first column is a label that marks the resample replicate during which the pseudoreplicate was generated. Only the first two replicates are shown.



	A	B	C	D	E
1	CURSAT ver 2.1 Edition #1	Date: 03-05-2020	Hour:	14:58:39	
2	Input file: incidence.txt				
3	Accumulation pseudoreplicates' file: accum_incidence_boot.txt				
4	Dataset pseudoreplicates' file: pseud_incidence_boot.txt				
5	Number of columns (sampling events): 10				
6	Number of rows (species, OTUs, ojects,...): 12				
7	Number of repetitions: 100				
8	Seed number: 12348695				
9					
10	Bootstrap (resampling with replacement)				
11					
12	Bootstrap replicate	Sampling event	Accumulation per n. of events		
13	1	1	6		
14	1	2	8		
15	1	3	11		
16	1	4	12		
17	1	5	12		
18	1	6	12		
19	1	7	12		
20	1	8	12		
21	1	9	12		
22	1	10	12		
23	2	1	7		
24	2	2	11		
25	2	3	11		
26	2	4	11		
27	2	5	12		
28	2	6	12		
29	2	7	12		
30	2	8	12		
31	2	9	12		
32	2	10	12		

Figure 3: Accumulation data file *accum_incidence_boot.txt* opened in Microsoft Excel. It consists of the output file with 100 accumulation replicates based on the bootstrap of the original dataset in the file *incidence.txt*. The first column marks the resample replicate during which the accumulation was generated. In the second and third columns, sampling events and the associated cumulative number of objects are respectively reported. Only the first two replicates are shown.

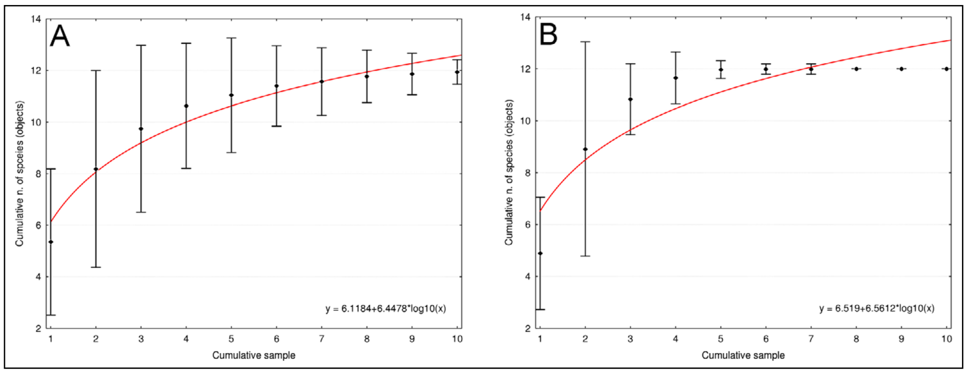


Figure 4: Accumulation curves (as indicated by black dots) obtained from 100 pseudoreplicates, with seed number = 12348695. The input file was *abundance.txt*. Bootstrap **(A)** and resampling without replacement **(B)** graphs. Dots indicate mean values. Whiskers indicate 2 × standard deviation. Only for reference purposes, logarithmic regression lines are drawn (red) and regression equations are reported. As expected, when using the same seed number, the files *abundance.txt* and *incidence.txt* produced identical results (not shown, but see also output files).

be imported in statistical packages to estimate distributions of successive pooling of samples and depict accumulation and rarefaction curves. The graphs in **Figure 4** were obtained using Statistica ver. 8 (StatSoft, Inc).

Quality control

CURSAT ver. 2.1 is written in basic (QB64). QB64 is not a cross-platform software *per se*. However, the QB64 programming language as released at https://www.qb64.

org/ is provided with stable builds in Microsoft Windows 32bit and 64bit, Linux, Apple Mac, and Chrome OS.

Instructions about how to install QB64 on Windows, Linux, and MacOS can be found at the website: https://www.qb64.org/wiki/QB64_FAQ#Q:_How_do_I_install_QB64_on_Windows.2C_Linux.2C_macOS.3F. Further instructions can be found in the README file associated with CURSAT ver. 2.1.

Once that QB64 has been installed on the desired Operating Systems, the source code (file with extension .bas) of CURSAT ver.2.1 can be loaded in QB64 environment and be compiled into an executable file that runs under the specific Operating Systems.

CURSAT ver. 2.1 was originally written using QB64 for Microsoft Windows 64bit. However, I also successfully compiled and run it using QB64 for Apple Mac under MacOS Mojave 10.14.5.

During the run, a bootstrap-replicate counter informs about the status of the run. Once the run has ended, the software informs the user and reports the names of files where output data are saved, along with information about the duration of the run. Files can be opened using a text editor or a spreadsheet and can be inspected for consistency and used for later analyses.

The software is provided in two editions. Edition #1 is recommended for analysis purposes. Edition #2 adds a log file (named by the user) where information used by the software (matrices D, ND, W, B, and F in the source code) generated and used during resampling and accumulation routines is stored and saved. Such information may prove useful for users to identify whether the software is operating as expected. Additionally, the possibility to check on intermediate steps of the software may also be useful for educational purposes.

Figure 5 illustrates in detail the way the program works. The source code of both editions is extensively commented. Each routine and relevant steps are anteceded by REM lines. Edition #2 has been used, along with vWATCH64 ver. 1.104, (https://www.qb64.org/vwatch/) for debugging.

Testing

The software is provided with example sample files, used for testing, and correspondent output files. The input file *abundance.txt* is a 12 × 10 rectangular matrix of abundances, whereas *incidence.txt* (see **Table 1**) is the incidence data matrix derived from *abundance.txt*. The file *data.txt* (as per the example in **Figure 5**) is also provided, along with output files *accum_data_shuf.txt*, *pseud_data_shuf.txt*, *data_shuf_log.txt*, and other files.

The database in the file *seedbank.txt* provided along with EstimateS ver. 9.1.0 [15] was also used for testing. The original file was slightly edited to meet the input requirements of CURSAT v.2.1. The edited file is here provided. The file consists of an abundance matrix with 34 rows and 121 columns. All input files were used to test CURSAT v.2.1. For each input file, 100 replicates were obtained by resampling with and without replacement. All runs were all performed with seed = 12348695. Results from using the database in the file *seedbank.txt* are shown in **Figure 6**.

Output files, including log files, as resulting from the different runs are provided along with the code and Windows executables. Multiple runs using the same seed number produced, as expected, the same results.

The software was also tested by using a 1000 × 1000 data matrix as in the file *1000.txt* provided as well, along with the code.

Such a matrix is an abundance matrix generated by Microsoft Excel in two steps. In the first step, a random incidence matrix 1000 × 1000 was created. In a second step, elements = 1 were replaced with integer numbers randomly chosen between 1 and 30.

In such a way, the expected accumulation curve for both resampling procedures should predict full saturation after 10 replicates, with the following mean values (rounded to the integer value) from replicate 1 through 10, respectively: 500, 750, 875, 938, 969, 984, 992, 996, 998, 999. One hundred replicates were generated for both bootstrap and shuffling. Accumulation curves are shown in **Figure 7**. Means observed and restricted standard deviations demonstrate that expectations were met in both cases (**Figure 7A** and **B**). For comparison, the same bootstrap analysis was also run by using EstimateS 9.1.0. CURSAT ver.2.1 and EstimateS 9.1.0 generated very similar results (**Figure 7C**).

(2) Availability

Operating system

The Windows executable file was successfully tested for functionality under Microsoft Windows 7 and 10 64 bit. The Apple Mac executable ran properly under MacOS Mojave 10.14.5.

Programming language

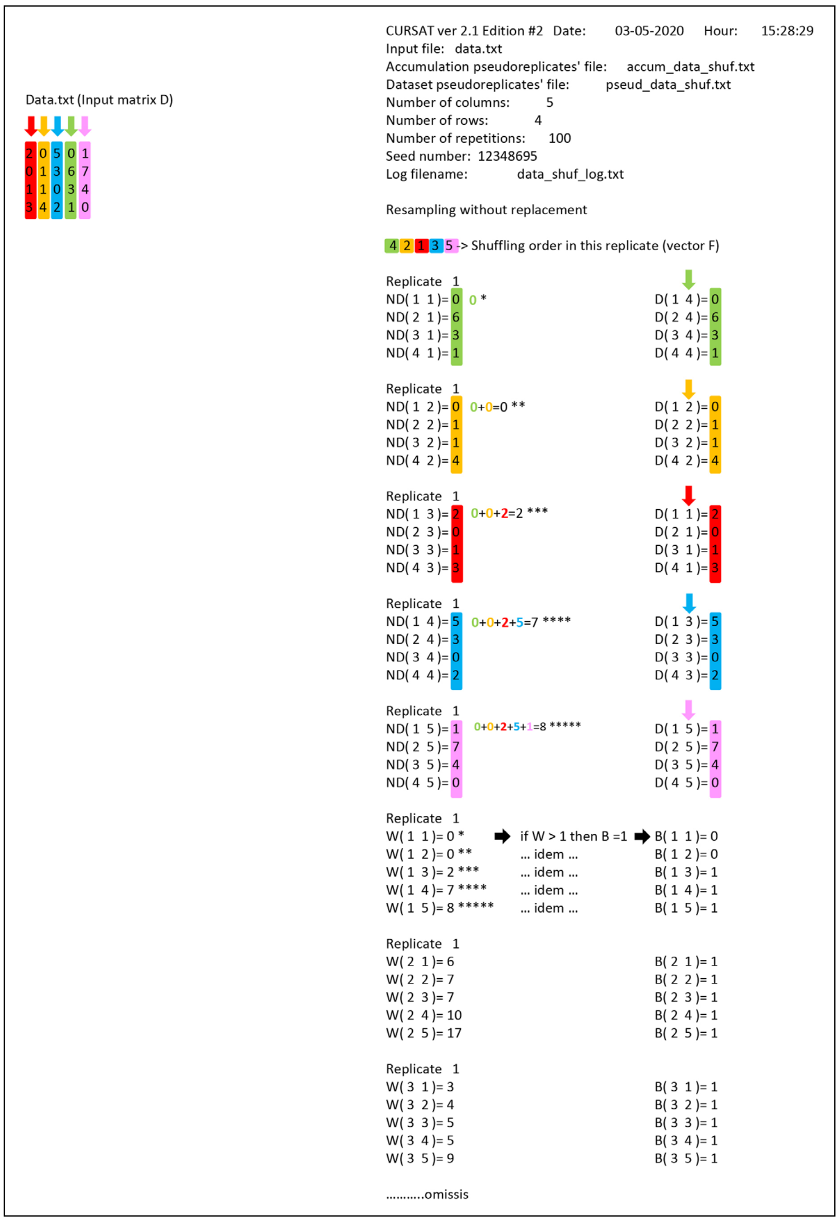
The source code of CURSAT ver. 2.1 is written in QB64 ver.0.960 basic. Syntax used by QB64 may slightly differ from other versions of the basic language (e.g. QB4.5, GWbasic), requiring some adjustments.

Additional system requirements

Once compiled into a Microsoft Windows 64 bit executable, and irrespective whether running random resampling with or without replacement, Edition #1 required less than 20000 Kb of memory and less than 1 minute to generate 10000 bootstrapped pseudoreplicates of a 12 × 10 incidence matrix, on an Intel® Core™ i7 CPU 860 at 2.80 Ghz, with 4.0 Gb RAM and Windows 7 64 bit. The size of the generated data and accumulation pseudoreplicates files was 3770 Kb and 865 Kb, respectively.

Edition #1 required approximately 10 seconds to generate 100 pseudoreplicates of a 100 × 100 data matrix. Such time increased to approximately 2 minutes and 20 seconds for Edition #2. In these last two cases, the size of pseudoreplicate and accumulation data files was 3975 Kb and 175 Kb, respectively. The log file generated by Edition #2 was 75034 Kb.

When using the abundance dataset in *1000.txt*, Edition #1 generated 2 files with 100 pseudoreplicate and accumulation data in 23 minutes, irrespectively whether bootstrap or shuffling were performed. Bootstrap and



```
Data.txt (Input matrix D)

2 0 5 0 1
0 1 3 6 7
1 1 0 3 4
3 4 2 1 0
```

```
CURSAT ver 2.1 Edition #2   Date:      03-05-2020    Hour:     15:28:29
Input file:   data.txt
Accumulation pseudoreplicates' file:     accum_data_shuf.txt
Dataset pseudoreplicates' file:        pseud_data_shuf.txt
Number of columns:         5
Number of rows:            4
Number of repetitions:      100
Seed number:  12348695
Log filename:            data_shuf_log.txt

Resampling without replacement

4 2 1 3 5 -> Shuffling order in this replicate (vector F)

Replicate   1
ND( 1  1 )= 0   0 *                          D( 1  4 )= 0
ND( 2  1 )= 6                                D( 2  4 )= 6
ND( 3  1 )= 3                                D( 3  4 )= 3
ND( 4  1 )= 1                                D( 4  4 )= 1

Replicate   1
ND( 1  2 )= 0   0+0=0 **                     D( 1  2 )= 0
ND( 2  2 )= 1                                D( 2  2 )= 1
ND( 3  2 )= 1                                D( 3  2 )= 1
ND( 4  2 )= 4                                D( 4  2 )= 4

Replicate   1
ND( 1  3 )= 2   0+0+2=2 ***                  D( 1  1 )= 2
ND( 2  3 )= 0                                D( 2  1 )= 0
ND( 3  3 )= 1                                D( 3  1 )= 1
ND( 4  3 )= 3                                D( 4  1 )= 3

Replicate   1
ND( 1  4 )= 5   0+0+2+5=7 ****               D( 1  3 )= 5
ND( 2  4 )= 3                                D( 2  3 )= 3
ND( 3  4 )= 0                                D( 3  3 )= 0
ND( 4  4 )= 2                                D( 4  3 )= 2

Replicate   1
ND( 1  5 )= 1   0+0+2+5+1=8 *****            D( 1  5 )= 1
ND( 2  5 )= 7                                D( 2  5 )= 7
ND( 3  5 )= 4                                D( 3  5 )= 4
ND( 4  5 )= 0                                D( 4  5 )= 0

Replicate   1
W( 1  1 )= 0 *       ➡  if W > 1 then B =1 ➡ B( 1  1 )= 0
W( 1  2 )= 0 **         ... idem ...         B( 1  2 )= 0
W( 1  3 )= 2 ***        ... idem ...         B( 1  3 )= 1
W( 1  4 )= 7 ****       ... idem ...         B( 1  4 )= 1
W( 1  5 )= 8 *****      ... idem ...         B( 1  5 )= 1

Replicate   1
W( 2  1 )= 6                                 B( 2  1 )= 1
W( 2  2 )= 7                                 B( 2  2 )= 1
W( 2  3 )= 7                                 B( 2  3 )= 1
W( 2  4 )= 10                                B( 2  4 )= 1
W( 2  5 )= 17                                B( 2  5 )= 1

Replicate   1
W( 3  1 )= 3                                 B( 3  1 )= 1
W( 3  2 )= 4                                 B( 3  2 )= 1
W( 3  3 )= 5                                 B( 3  3 )= 1
W( 3  4 )= 5                                 B( 3  4 )= 1
W( 3  5 )= 9                                 B( 3  5 )= 1

..........omissis
```

Figure 5: An example of a log file generated by running Edition #2, illustrating in detail the algorithm used by CURSAT ver. 2.1. The input data matrix D (top-left) in the file *data.txt* was used in this example. Only outcomes of replicate n.1 from a shuffling procedure are shown because the procedure is very similar for bootstrap. For each resampling replicate, a pseudoreplicate data matrix (ND) is generated by extracting the elements from a column of the input matrix D, according to the shuffling order stored in vector F. In this example, elements in column #1 of ND are extracted from column #4 of the input matrix D (green); subsequently, elements in column #2 of ND are extracted from column #2 of the input matrix D (orange), and so on. A new matrix (W) is then constructed by cumulatively summing elements of the pseudoreplicate data ND matrix by row (see asterisks). Finally, the matrix B (same dimensions as W) is constructed from matrix W, by replacing elements >0 with 1. This is instrumental to correctly perform accumulation irrespectively whether the input file contains abundance (as in this case) or incidence data. The accumulation replicates are constructed by cumulatively summing elements of matrix B by column (see files *data_shuf_log.txt* and *accum_data_shuf.txt*). The accumulation replicates are not stored in memory, but they are printed in the output file meanwhile they are created.

shuffling were conducted simultaneously on the same computer. Both procedures generated output files of 1.9 Kb and 42 Mb for pseudoreplicate and accumulation data, respectively.

A 1000 × 1000 data matrix was the largest data set used for the speed test of CURSAT ver.2.1. The software can handle larger matrices of data, but this is time-consuming.

Dependencies

No library is required. No previous knowledge of the BASIC language or QB64 environment is required to run CURSAT ver. 2.1, as the executable file, can run as "stand-alone" software. No official support is provided, but users are welcome to contact the author for trouble shooting.

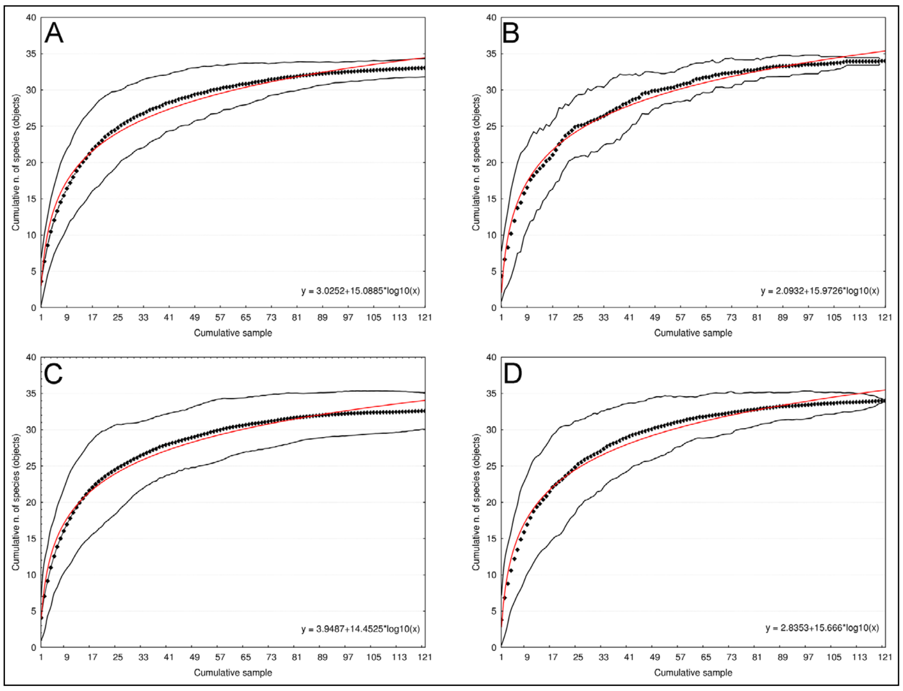


Figure 6: Accumulation curves (black dots) obtained from 100 pseudoreplicates. The database as in the file *seedbank.txt* was used. CURSAT ver. 2.1 **(A)** and **(B)** and EstimateS 9.1.0 **(C)** and **(D)** were tested for bootstrap and shuffling. Bootstrap and shuffling graphs are on the left and right sides, respectively. Dots indicate mean values. Standard deviation (2×) is indicated by continuous black lines. Logarithmic regression lines are drawn (red) and regression equations reported. CURSAT ver. 2.1 and EstimateS 9.1.0 produced similar results. For EstimateS 9.1.0 the cumulative n. of species (objects) has been calculated using S Mean ± (2 × bootstrap/shuffling SD (runs)) from the EstimatesS 9.1.0 output.

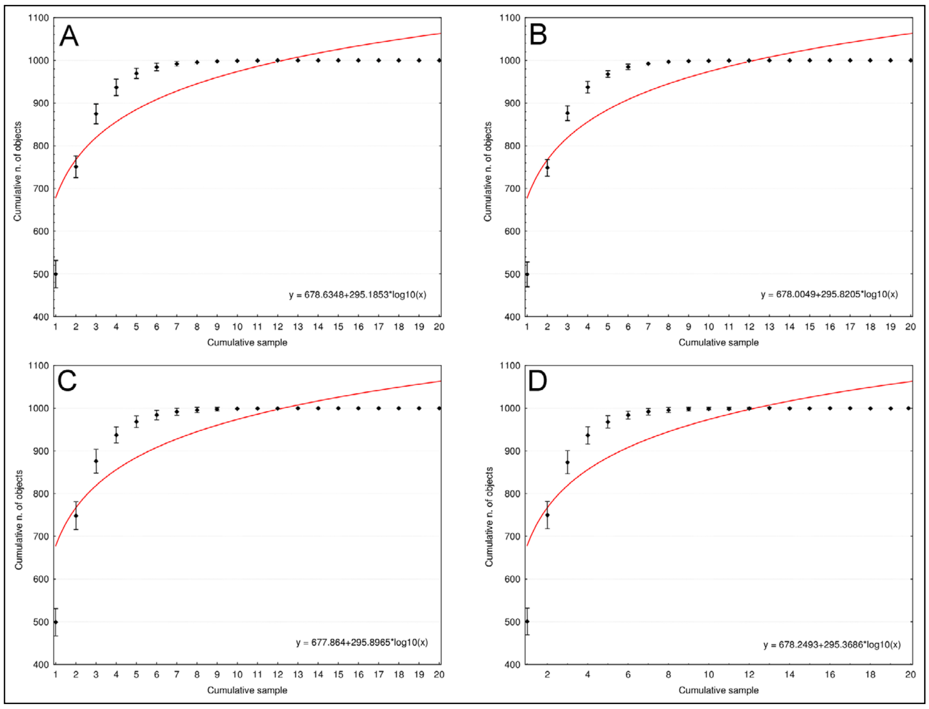


Figure 7: Accumulation curves (black dots) obtained from 100 pseudoreplicate data. The input data was the abundance 1000 × 1000 matrix as in the file *1000.txt*. CURSAT ver. 2.1 **(A)** and **(B)** and EstimateS 9.1.0 **(C)** and **(D)** were tested for bootstrap and shuffling. Bootstrap and shuffling graphs are on the left and right sides, respectively. Dots indicate mean values. Standard deviation (2×) is indicated by whiskers. Logarithmic regression lines are drawn (red) and regression equations are shown. To improve clarity, only cumulative data the first 20 sampling events are here reported. As expected in the case of a random matrix, both bootstrap and shuffling produced almost the same results. For EstimateS 9.1.0 the cumulative n. of objects has been calculated using S Mean ± (2 × bootstrap/shuffling SD (runs)) from the EstimatesS 9.1.0 output.

List of contributors

The software was entirely created by the author.

Software location

Archive

Name: Zenodo
Persistent identifier: 10.5281/zenodo.3700081
Licence: MIT License
Publisher: Gabriele Gentile
Version published:
Date published: 06/03/2020

Code repository

Name: GitHub
Identifier: https://github.com/gabrio62/CURSAT-ver.2.1
Licence: MIT License
Date published: 06/03/2020

Language

English

(3) Reuse potential

The software can be used by ecologists, biogeographers, zoologists, botanists, conservationists but it also has general applicability in all cases in which data matrices from repeated sampling have to be evaluated by a resampling/rarefaction approach. For example, CURSAT ver. 2.1 has been recently used to account for a possible bias in the number of haplotypes caused by unequal sample size, in a genetic study of hemoparasites of Galápagos iguanas (Fulvo et al., submitted).

CURSAT ver. 2.1 can be used to perform random resampling with and without replacement of every kind of rectangular matrix. Thus, the possibility to extract pseudoreplicates of original data and easily import them into a spreadsheet allows calculating several different kinds of statistics and their associated variance. Variance estimated by a random sampling approach with replacements may provide a valid alternative in case the analytical derivation of asymptotic distribution and variance is not possible.

Edition #2 may be used to illustrate the internal steps of the software. This can be useful for educational purposes.

Following the suggestion of one of the reviewers a command-line version to facilitate the integration of such a program in a pipeline could be implemented in a more advanced version of the software, along with a batch mode allowing the user to run multiple files on the same run.

Additional Files

The additional files for this article can be found as follows:

- **Supplementary file 1.** CURSAT v.2.1 zipped package including README, codes, example files. DOI: https://doi.org/10.5334/jors.260.s1
- **Supplementary file 2.** Input and Output files in rar archive. DOI: https://doi.org/10.5334/jors.260.s2
- **Supplementary file 3.** CURSAT ver.2.1 Edition 1 source code. DOI: https://doi.org/10.5334/jors.260.s3
- **Supplementary file 4.** CURSAT ver.2.1 Edition 2 source code. DOI: https://doi.org/10.5334/jors.260.s4
- **Supplementary file 5.** CURSAT ver.2.1 Edition 1 executable. DOI: https://doi.org/10.5334/jors.260.s5
- **Supplementary file 6.** CURSAT ver.2.1 Edition 2 executable. DOI: https://doi.org/10.5334/jors.260.s6
- **Supplementary file 7.** Readme file for CURSAT ver. 2.1. DOI: https://doi.org/10.5334/jors.260.s7

Acknowledgements

I am grateful to a very special person who provided me a Commodore64 to learn basic programming language.

Competing Interests

The author has no competing interests to declare.

References

1. **Ridenhour, J** and **Grimmett, D** 2006 Sampling and Related Binomial Identities. *The College Mathematics Journal*, 37: 296–299. DOI: https://doi.org/10.2307/27646360
2. **Hill, M O** 1973 Diversity and evenness: a unifying notation and its consequences. *Ecology*, 54: 427–432. DOI: https://doi.org/10.2307/1934352
3. **Chao, A** 1984 Nonparametric estimation of the number of classes in a population. *Scandinavian Journal of Statistics*, 11: 265–270.
4. **Chao, A** and **Jost, L** 2015 Estimating diversity and entropy profiles via discovery rates of new species. *Methods in Ecology and Evolution*, 6: 873–882. DOI: https://doi.org/10.1111/2041-210X.12349
5. **Chao, A** 1987 Estimating the population size for capture-recapture data with unequal catchability. *Biometrics*, 43: 783–791. DOI: https://doi.org/10.2307/2531532
6. **Colwell, R K, Mao, C X** and **Chang, J** 2004 Interpolating, extrapolating, and comparing incidence-based species accumulation curves. *Ecology*, 85: 2717–27. DOI: https://doi.org/10.1890/03-0557
7. **Colwell, R K, Chao, A, Gotelli, N J, Lin, S-Y, Mao, C X, Chazdon, R L** and **Longino, J T** 2012 Models and estimators linking individual-based and sample-based rarefaction, extrapolation and comparison of assemblages. *Journal of Plant Ecology*, 5: 3–21. DOI: https://doi.org/10.1093/jpe/rtr044
8. **Gotelli, N J** and **Colwell, R K** 2011 Estimating species richness. In Magurran, A E and McGill, B J (eds.), *Biological diversity: frontiers in measurement and assessment*, 39–54. New York, USA: Oxford University Press.
9. **Chao, A, Gotelli, N J, Hsieh, T C, Sander, E L, Ma, K H, Colwell, R K** and **Ellison, A M** 2014 Rarefaction and extrapolation with Hill numbers: a framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84: 45–67. DOI: https://doi.org/10.1890/13-0133.1
10. **Colwell, R K** 2013 EstimateS 9.1.0 User's Guide, http://viceroy.colorado.edu/EstimateS/EstimateSPages/EstSUsersGuide/EstimateSUsersGuide.

htm#WhatEstimateSComputes. Accessed on February 2nd 2020.

11. **Egan, A N** and **Crandall, K A** 2006 Theory of Phylogenetic Estimation. In Fox, C W and Wolf, J B (eds.), *Evolutionary Genetics: Concepts and Case Studies*, 426–444. London: Oxford University Press.
12. **Benítez-Malvido, J, Dáttilo, W, Martínez-Falcón, A P, Durán-Barrón, C, Venezuela, J, López, S** and **Lombera, R** 2016 The multiple impacts of tropical forest fragmentationon arthropod biodiversity and on their patterns of interactions with host plants. *PloSOne*, 11: 1–15. DOI: https://doi.org/10.1371/journal.pone.0146461
13. **Scotti, I, Montaigne, W, Cseke, K** and **Traissac, S** 2013. RaBoT: a rarefaction-by-bootstrap method to compare genome-wide levels of genetic diversity. *Annals of Forest Science*, 70: 631–635. DOI: https://doi.org/10.1007/s13595-013-0302-z
14. **Oksanen, J, Blanchet, F G, Kindt, R, Legendre, P, O'Hara, R B, Simpson, G L, Solymos, P, Stevens, M H H** and **Wagner, H** 2010 Vegan: community ecology package. *R package version 1.17-4*. http://cran.r-project.org.
15. **Schloss, P D, Westcott, S L, Ryabin, T, Hall, J R, Hartmann, M, Hollister, E B, Lesniewski, R A, Oakley, B B, Parks, D H, Robinson, C J, Sahl, J W, Stres, B, Thallinger, G G, Van Horn, D J** and **Weber, C F** 2009 Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and Environmental Microbiology*, 75: 7537–7541. DOI: https://doi.org/10.1128/AEM.01541-09
16. **Colwell, R K** and **Elsensohn, J E** 2014 EstimateS turns 20: statistical estimation of species richness and shared species from samples, with non-parametric extrapolation. *Ecography*, 37: 609–613. DOI: https://doi.org/10.1111/ecog.00814
17. **Hsieh, T C, Ma, K H** and **Chao, A** 2016 iNEXT: An R package for rarefaction and extrapolation of species diversity (Hill numbers). *Methods in Ecology and Evolution*, 7: 1451–1456. DOI: https://doi.org/10.1111/2041-210X.12613
18. **Saary, P, Forslund, K, Bork, P** and **Hildebrand, F.** 2017 RTK: efficient rarefaction analysis of large datasets. *Bioinformatics*, 33: 2594–2595. DOI: https://doi.org/10.1093/bioinformatics/btx206

How to cite this article: Gentile, G 2020 CURSAT ver. 2.1: A Simple, Resampling-Based, Program to Generate Pseudoreplicates of Data and Calculate Rarefaction Curves. *Journal of Open Research Software,* 8: 17. DOI: https://doi.org/10.5334/jors.260

Submitted: 22 February 2019 **Accepted:** 25 June 2020 **Published:** 21 August 2020