recon_v3.0.py

Reconstruction of Estimated Communities from Observed Numbers

Joseph Kaplinsky, PhD and Ramy Arnaout, MD, DPhil

April 20, 2016
revised July 10, 2020

0. Contents

1. Introduction

1.1. Overview

Recon is an algorithm for generating a description of an overall population from a sample (see section 1.2, Terminology). More precisely, Recon uses the distribution of species counts in a sample to estimate the distribution of species counts in the population from which the sample was drawn. This is useful for large and complex populations, in which samples are likely to misrepresent the distribution of species in the overall population.

1.2. Terminology

A *species* is group (class) made up of one or more *individuals*. The number of individuals of a given species is that species' *size*. A *sample* is a set of individuals drawn randomly from an *overall* or *parent* population. Species represented in the population that are not represented in the sample are called *missing species*. (For historical reasons, in the code and below, *clone* is used interchangeably for *species*.)

Diversity refers to any of a set of measures of the frequency distribution in the population. These measures can be thought of as effective numbers of species in the population. The *Hill numbers* (${}^{q}D$; read e.g. "D-one" for ${}^{1}D$ and "D-infinity" for ${}^{\infty}D$) are a family of diversity measures defined by the parameter *q*, which determines the degree to which diversity measures are weighted toward larger species. Many Hill numbers correspond to familiar measures. For example, ${}^{0}D$ is *species richness*, a diversity measure that weights all species equally (and so is just a count of the number of different species present, regardless of their relative frequencies). ${}^{\infty}D$ is the reciprocal of the Berger-Parker Index, the effective number of species if all species were the size of the largest species. Simple mathematical transformations of ${}^{1}D$ and ${}^{2}D$ correspond to Shannon entropy (information theory) and the Gini-Simpson Index (economics), respectively.

1.3. Features

Key features of Recon include that it (i) assumes nothing about the shape of the frequency distribution of species in the population (e.g. exponential, power, arbitrary, etc.), (ii) avoids (over)fitting of sampling noise, (iii) scans many starting points in an attempt to find a global best fit, (iv) outputs 95% confidence intervals for the number of species of a given size in the population, and (v) outputs power tables useful in designing experiments for comparing overall populations based on samples.

Please see the license (Section 10) for rules of use.

1.4. Citing Recon

Please cite Recon as follows:

Kaplinsky J. and Arnaout R.(2016) Robust Estimates of Overall Immune-Repertoire Diversity from High-Throughput Measurements on Samples. *Nat. Communications*, In Press.

2. Installation

2.1. Availability

Recon is available on GitHub (https://github.com/ArnaoutLab/Recon) subject to the terms in the license (Section 10). It is contained in the file `recon_v3.0.py` (although note the requirements in section 2.2 below).

2.2. Requirements

Producing fits, diversity measures, and power tables requires:

- Python 3.6 or later (https://www.python.org)
- The SciPy Python library (http://scipy.org)

Producing plots additionally requires:

- d3.js (http://d3js.org)
- wkhtmltopdf (http://wkhtmltopdf.org)
- cpdf (http://community.coherentpdf.com)
- the file `style.css` (included in the Recon distribution)
- the JavaScript file `plot_clone_size_distribution_ref.js` (also included in the Recon distribution)

2.3. Platforms

Recon has been tested on Macintosh OS X (v10.8-11) and several other Unix systems.

2.4. Latest version

The latest version as of this writing is 2.2 (`recon_v3.0.py`)..

3. Operation

This section describes the various modes for running Recon. The main use is to produce a description of an overall population from a sample (section 3.1). it can be used to build an error-bar profile (section 3.2; necessary for calculating error bars and power tables), output diversity measures (section 3.3), produce power tables (section 3.4), and generate plots (section 3.5).

3.1. Reconstruction (`-R, --run_recon`)

3.1.1. Description

Given a set of observations of species frequencies in a sample as input, the -R option outputs a set of parameters that describe the modified maximum-likelihood species frequency distribution in the parent population (see below), without overfitting.

3.1.2. Usage

`python recon_v3.0.py -R` *[options...]* `-o OUTPUT_FILE INPUT_FILE`

3.1.3. Input

Input is a text file containing the number of individuals of each species seen in the sample (i.e., the species sizes in the sample). The name of this file is supplied as the parameter `INPUT_FILE`.

The default format is a tab-delimited file with lines of the form

species name <tab> species size

with a newline character delimiting lines. Species size is an integer. Sample data in this format can be found in the file `test_sample_4.txt`; in it, species 9_0 has a size of 9 individuals, individual, species 8_0 has 8 individuals, species 1_1833458 has 1 individual, etc.

An alternative format is specified by the `-c` (`--clone_distribution_in_file`) option (see section 3.1.6 below). This is a tab-delimited format with lines of the form

species size <tab> number of species of this size

where both values are integers. `test_sample_1.txt`, `test_sample_2.txt`, and `test_sample_3.txt` are all in this format. The file `test_sample_4.txt` contains the same data as `test_sample_1.txt`, only in the default extended format.

Note that files in the default format can be long and therefore take a few seconds to read (`test_sample_4.txt` is over two million lines long). The alternative format is much more compact and therefore faster to read. For example, `test_sample_1.txt` contains the same information as `test_sample_4.txt`, only in this more compact format, and runs in ~2 seconds vs. ~7 seconds for `test_sample_4.txt`.

3.1.4. Output

The above command returns a tuple with the following elements, in order:

- a list of weights that, with means, describes the reconstructed parent distribution. Each weight w_i is the fraction of all species in the parent distribution that each contribute a mean m_i number of individuals to the sample;
- a list of means that, with weights, describes the reconstructed parent distribution. Eacn mean m_i is the mean number of individuals a species of this size contributes to the sample. Means are Poisson parameters;
- an integer of the number of missing species;
- a dictionary of the the species-size distribution in the sample (the observed distribution), where each key is a species size, and the corresponding value is the number of species of that size. If the `-c` option is given, the keys and values should correspond to the left and right columns of input data (a useful check that your data was read in successfully);
- if given (the `-A` option), the true number of species in the parent population;
- a float of the log-likelihood of this fit; and
- a float of the time in seconds for the fit.

See 3.1.6 below for an example.

Recon also writes to `FILE_OUT` a summary of the fit. We refer to files of this type as “fitfiles;” fitfiles are used as input for other Recon functions (below). The last block in the fitfile that is offset by multiple equal signs (“`=======`“) contains the final weights and means (as a single list of weights-then-means; denoted by lines with “`fitted parameters`”) and missing species (denoted by lines with “`estimated n0`”). These are the same as in the screen output described above.

3.1.5. Required parameters

`-R, --run_recon`	Required to run Recon in this mode (i.e. to tell it to reconstruct an overall distribution)
`-o OUTPUT_FILE`	The filename to be used for output. Note that if this file exists it will not be overwritten; instead recon will exit with an error message.
`INPUT_FILE`	A text file containing the number of individuals of each species seen in the sample (i.e. the species sizes in the sample). See 3.1.4 above for possible formats for this file.

3.1.6. Optional parameters

`-a, --aicc_multiple AICC_MULTIPLE`	Sets the multiple of the observed number of datapoints that Recon considers observations. E.g., if only singletons, doublets, and triplets are observed but user believes not seeing quadruplets is

evidence of absence (vs. absence of evidence), `-a 1.3` will tell Recon to consider this as four observations for purposes of calculating AICc.

`-c, --clone_distribution_in_file`

Allows recon to read an alternative tab-delimited format with lines of the form

species size <tab> number of species of this size

where both values are integers. test_sample_1.txt, test_sample_2.txt, and test_sample_3.txt are all in this format. As mentioned above, test_sample_4.txt contains the same data as test_sample_1.txt, only in the default extended format.

`-d, --bin_size BIN_SIZE`

Average number of observations per individual (default: 1). In many circumstances, each individual in the sample will be observed and counted once. However, there are cases where each individual in the sample will be observed and counted multiple times. `BIN_SIZE` allows for this possibility.

`-l, --parameter_limit PARAMETER_LIMIT`

The maximum number of parameters that the algorithm will use to fit the data (default, 20). Recon will continue adding parameters until the AICc indicates that additional parameters are not justified. In practice the limit of 20 is essentially never reached.

`-s, --noise_test_ratio_threshold NOISE_TEST_RATIO_THRESHOLD`

Sets the noise threshold ratio. Recon will reject fitting a population of clones that is so small that its contribution to the sample is comparable to the sampling noise from other populations that are present.

`NOISE_TEST_RATIO_THRESHOLD` is the factor by which the expected contribution of a population of clones must rise above the sampling noise (i.e. standard deviation from Poisson sampling in the contribution to the count of singletons) from the next largest population. Empirically, the default of 3.0 produces good results for fitting distributions of the sort reported in the paper (see section 1.4).

The noise threshold ratio corresponds to a false-positive rate for fitting a clone population to sampling noise. If the noise threshold ratio is set higher than 3.0 then Recon will be less sensitive to noise (lower false positive rate), but will fit fewer missing species (higher false negative rate). Conversely, if it is set lower, it will be more sensitive to noise (higher false positive rate), but fit more missing species (lower false negative rate).

The exact false poitive rate that 3.0 corresponds to depends on the mean parameter of the smallest fitted clone size. The false positive rate for accepting the presence of a small clone size can be found by considering the CDF of a Poisson distribution. Specifically, it will be

gammaincc(NOISE_TEST_RATIO_THRESHOLD**m*, *m*), where gammaincc is the complementary incomplete gamma function and *m* is mean parameter of the smallest fitted clone size (see section 3.1.7 for some example means).

At a noise threshold ratio of 3.0, this gives:

m	false positive rate
1e-1	0.45
1e-2	0.11
1e-3	1.8e-2
1e-4	2.6e-3
1e-05	3.3e-4
1e-06	4.0e-05

Typical values of *m* for which noise is a limiting factor are 1e-3 or below. At a noise threshold ratio of 3.0 this therefore corresponds to false positive rates of less than 2%.

`-t --threshold THRESHOLD`

Allows you to modify THRESHOLD, the clone size above which sampling error is considered small, which means that Recon will assume the frequency of species of this size or greater in the population is the same as the frequency in the sample. It defaults to 30, which usually gives good results: this is because if, in a sample from a well mixed population, species A is seen 30 times in a sample it is very unlikely that there is another species B which is the same size as A in the parent population but is missing or very poorly represented in the sample. Smaller values will give faster run times but less accurate results.

3.1.7. Example

Command:

```
python recon_v3.0.py -R -t 30 -c -o test_sample_1_fitfile.txt
test_sample_1.txt
```

Output:

```
([0.88205137420509439, 0.11794862579490561], [0.306026605847127,
1.0723167141789034], 5224621, {1: 1833459, 2: 405423, 3: 86822,
4: 18467, 5: 3694, 6: 626, 7: 128, 8: 20, 9: 1}, None, -
1572523.5668443954, 2.2577288150787354)
```

As described above, this output consists of:

list of weights	`[0.88205137420509439, 0.11794862579490561]`
list of means	`[0.306026605847127, 1.0723167141789034]`
number of missing species	`5224621`
observed (input) distribution	`{1: 1833459, 2: 405423, 3: 86822, 4: 18467, 5: 3694, 6: 626, 7: 128, 8: 20, 9: 1}` Note this means that in the observed (input) data, 1,833,459 species were represented by a single individual each, 405,423 species were represented by two individuals each, and so on.
true number of species in the parent population	`None` (meaning none was given; this would have been supplied as part of the command using the –a option)
log-likelihood of this fit	`-1572523.5668443954`
seconds elapsed	`2.2577288150787354`

3.2. Error bars (**`-e, --make_error_bars`**)

3.2.1. Description

This mode generates an error bar parameter file from a set of fits on data for which the number of missing species is known (i.e., validation datasets). This file is needed for D-number tables (section 3.3) and power tables (section 3.4).

3.2.2. Usage

```
python recon_v3.0.py -e -o OUTPUT_FILE ERROR_BAR_FIT_DIRECTORY
```

3.2.4. Input

A directory that contains fits for distributions with known numbers of species (and therefore known numbers of missing species).

3.2.4. Output

An error bar parameter file.

3.2.5. Required parameters

`-e, --make_error_bars`	Tells recon to make an error-bar parameter file.
`-o, --file_out OUTPUT_FILE`	The name of the new error-bar parameter file
`ERROR_BAR_FIT_ DIRECTORY`	The name of a directory that contains the fits with known missing species. (The known missing species are encoded in the weights and means of the population.)

3.2.6. Example

```
python recon_v3.0.py -e error_bar_parameters.txt Test_dir
```

3.3. Diversity measures (`-D, --make_table_of_D_numbers`)

3.3.1. Description

Given a fitfile (either the output from section 3.1 above or any population description in that format), outputs a table of diversity measures as Hill numbers (see section 1.2, Terminology). Note that measures for any Hill number are obtainable, but the appropriate Hill number(s) must have been provided when making the `ERROR_BAR_PARAMETER_FILE` that Recon uses for outputting diversity (section 3.2 above).

3.3.2. Usage

`python recon_v3.0.py -D` *[options...]* `-b ERROR_BAR_PARAMETER_FILE -o OUTPUT_FILE INPUT_FILE` *[INPUT_FILE_2 INPUT_FILE_3 ...]*

3.3.3. Input

`INPUT_FILE`	A fitfile, which can be either the output from section 3.1 above or any population description in that format
`ERROR_BAR_PARAMETER_ FILE`	A text file that Recon uses to calculate error bars produced by the `-e` (`--make_error_bars`) option

3.3.4. Output

A table of Hill numbers for the reconstructed distribution, one row for each input file (see `INPUT_FILE` below). Columns prefixed "`obs_`" show the Hill numbers from the observed data in the sample (sample diversities). Columns prefixed "`est_`" show the Hill numbers Recon has estimated for the population (overall diversities). The difference between `est_0D` and `obs_0D` is n_0, the estimated number of missing species. Columns with the suffixes "+" and "−" indicate upper and lower error-bar limits, respectively. If no error-bar parameters exist in the error-bar parameter file for (one of) the value(s) of *q* that is given, the upper and lower error-bar limits will be blank.

3.3.5. Required parameters

`-D, --make_table_of_D_numbers`	Tells Recon to make a table of D numbers
`-b ERROR_BAR_PARAMETERS`	`ERROR_BAR_PARAMETERS` is a file that contains parameters for constructing error bars on fits. The supplied file `error_bar_params.txt` can be used. Alternatively, Recon can generate a custom error-bar parameter file from a set of gold-standard fits (see section 3.2 above).
`-o, --file_out OUTPUT_FILE`	The desired name of the output file.
`INPUT_FILE [INPUT_FILE_2 INPUT_FILE_3 ...]`	The input file(s). Multiple files should be separated by spaces; each will generate one rown in the output table. Each input file is fitfile (see section 3.3.3).

3.3.6. Optional parameters

`-Q HILL_NUMBER [HILL_NUMBER_2 HILL_NUMBER_3 ...]`	Hill-number parameter(s) for table (i.e., *q*). Note that the error-bar parameters file must be run for whatever Hill numbers are desired for the power table. Multiple parameters should be separated by spaces.

3.3.7. Example

```
python recon_v3.0.py -D -Q 0 1 inf -b error_bar_parameters.txt –o
test_D_number_table.txt test_sample_1_fitfile.txt
test_sample_2_fitfile.txt test_sample_3_fitfile.txt
```

This will output a file called `test_D_number_table.txt` with a header containing some reference information:

```
# python recon_v3.0.py -D -Q 0 inf -b error_bar_parameters.txt -o
test_D_number_table.txt test_sample_1_fitfile.txt test_sample_2_fitfile.txt
test_sample_3_fitfile.txt
# infiles = ['test_sample_1_fitfile.txt', 'test_sample_2_fitfile.txt',
'test_sample_3_fitfile.txt']
```

```
# observed_threshold = 30
# precomputed_error_bar_file = error_bar_parameters.txt
```

and the following lines (for clarity, only the first five columns are shown here and the values are have been truncated to two decimal points):

```
sample_name               obs_0.0D     est_0.0D     est_0.0D-    est_0.0D+
test_sample_1_fitfile.txt 2348640.0    7573260.81   5755678.21   inf
test_sample_2_fitfile.txt 2500.0       3013.53      2709.16      3513.20
test_sample_3_fitfile.txt 336.0        472.77       412.73       580.95
```

3.4. Power tables (`-p, --make_power_table`)

3.4.1. Description

Generates a power table with the minimum sample size required to be able to detect differences of a given magnitude in a given Hill number for two populations. That is, if you have two populations, and want to be able detect a difference in ${}^1D$ of x%, the table tells you how big your samples have to be.

3.4.2. Usage

`python recon_v3.0.py -p` *[-q HILL_NUMBER -m MIN_NUMBER_OF_DOUBLETS]* `-o FILE_OUT ERROR_BAR_PARAMETERS`

3.4.3. Input

An error-bar parameters file (`ERROR_BAR_PARAMETERS`) and the name of the output file that will contain the power table.

3.4.4. Output

A power table in which the rows and columns show the minimum fold differences that can be detected for different population sizes. Recon produces a single power table for each Hill number, so building tables for e.g. q=0, q=1, and q=2 requires running Recon with the `-p` option separately with `-q 0` (the default), `-q 1`, and `-q 2`.

If the output file already exists, Recon will issue a warning and print the output to stdout without overwriting the existing file.

3.4.5. Required parameters

`-o, --file_out OUTPUT_FILE`	The desired name of the output file.
`ERROR_BAR_PARAMETERS`	A file that contains parameters for constructing error bars on fits. The supplied file `error_bar_params.txt` can be used. Alternatively, Recon can generate an error bar parameter file from a set of gold standard fits (see section 3.3).

3.4.6. Optional parameters

`-C NUMBER_OF_SPECIES [NUMBER_OF_SPECIES_2 NUMBER_OF_SPECIES_3…]`	The (space-delimited) rough number of species in the overall population to consider; used as the columns of the power table. Default is `1e4 3e4 1e5 3e5 1e6 3e6` (10,000, 30,000, 100,000, 1 million, 3 million).
`-F FOLD_DIFFERENCES [FOLD_DIFFERENCES_2 FOLD_DIFFERENCES_3…]`	The rows of the table. Default is `1.1 1.2 1.3 1.4 1.5 2.0 5.0` (i.e., 10%, 20%, 30%, 40%, 50%, 2x, and 5x).
`-q, --q HILL_NUMBER`	The Hill number for which the power calculation is carried out (default: 0)
`-m, --min_number_of_ doublets MIN_NUMBER_OF_DOUBLETS`	An additional statistical minimum required for good results. The default of 100 should be good for most purposes.

3.4.7. Example

```
python recon_v3.0.py -p -C 1e4 1e5 1e6 -F 1.5 5 10 -q 1 -o
test_power_table.txt error_bar_parameters.txt
```

This outputs a file called `test_power_table.txt` with the following contents (header information followed by the actual table):

```
# Wed Apr 20 16:10:29 2016
# python recon_v3.0.py -p -C 1e4 1e5 1e6 -F 1.5 5 10 -q 1 -o
test_power_table.txt error_bar_parameters.txt
error_bar_file = test_power_table.txt
number_of_error_bars = 2.0
min_number_of_doublets = 100
fraction_small_clones = 0.1
q = 1

        10000     100000     1000000
1.5     14142     44721      141421
5.0     14142     44721      141421
10.0    14142     44721      141421
```

3.5. Resampling (**`-r, --resample`**)

3.5.1. Description

This allows resampling of a fit (model). The output gives the maximum likelihood observed species size distribution of samples from the model. This is the distribution that Recon attempts to make as close as possible to the observed species size distribution. The closeness of the fits can be compared to measure the goodness of fit.

3.5.2. Usage

```
python recon_v3.0.py -r -o OUTPUT_FILE INPUT_FILE
```

3.5.3. Input

A Recon fit file such as those produced by the `-R` option (see section 3.1).

3.5.4. Output

A list of species sizes up to the threshold that was used in the original fit is outputted, together with a count of species for each size. Output is written to `OUTPUT_FILE`.

3.5.5. Required parameters

`OUTPUT_FILE`	The desired name of the output file.
`INPUT_FILE`	A file that contains model fitted parameters, for example as output from a previously completely Recon fit (see section 3.1)

3.5.6. Example

```
python recon_v3.0.py -r -o test_sample_1_resample.txt
test_sample_1_fitfile.txt
```

The output file will consist of two (tab-delimited) colums with a headerline preceded by a hash tag (#). The first column lists the clone size, and the second column the number of clones:

```
#clone_size   no_clones
1             1833117
2             406083
3             86315
4             18638
5             3722
6             651
7             99
8             13
9             2
```

Note how this resampled distribution compares to the observed data, which can be found in the third line of the input file, `test_sample_1_fitfile.txt`:

```
observed_clone_size_distribution = {1: 1833459, 2: 405423, 3:
86822, 4: 18467, 5: 3694, 6: 626, 7: 128, 8: 20, 9: 1}
```

In columnar form:

```
1             1833459
2             405423
3             86822
```

```
4               18467
5               3694
6               626
7               128
8               20
9               1
```

one can see close agreement between the observed data and the resampled data, supporting this being a good fit. However, comparison is even easier with plots (the `-x` mode; section 3.6 below).

3.6. Plotting (**`-x, --make_resample_plot`**)

3.6.1. Description

Plots a resampled fit (see section 3.5) against the input data. Note the additional requirements for plotting (see section 2.2 above). Note clones larger than the threshold used for the fit (see `-t, --threshold` option in section 3.1.6) will still be plotted, but because these are data that have been added back in, these points will usually be spot-on.

3.6.2. Usage

```
python recon_v3.0.py -x [options...] -o PLOT_FILE -b
ERROR_BAR_PARAMETER_FILE INPUT_FILE
```

3.6.3. Input

A Recon fitfile and an error-bar parameter file.

3.6.4. Output

A plotfile containing the data to be plotted, and a PDF of the plot. These are described in more detail in section 3.6.7, below.

3.6.5. Required parameters

`-o PLOT_FILE`	The desired name of the output file that will contain the data to be plotted.
`-b ERROR_BAR_PARAMETER_FILE`	A file that contains parameters for constructing error bars on fits. The supplied file `error_bar_params.txt` can be used. Alternatively, Recon can generate an error bar parameter file from a set of gold standard fits (see section 3.3).

3.6.6. Optional parameters

`-X, --x_max X_MAX`	The upper limit of the x-axis in the plot. The default is set dynamically based on the data, up to a limit of `40`. (The user

	can override this limit using this option.)
`-Y, --y_max Y_MAX`	The upper limit of the y-axis in the plot. The default is set dynamically based on the data.
`-q, --q HILL_NUMBER`	The Hill number for which the power calculation is carried out (default: 0)
`-m, --min_number_of_doublets MIN_NUMBER_OF_DOUBLETS`	An additional statistical minimum required for good results. The default of 100 should be good for most purposes.

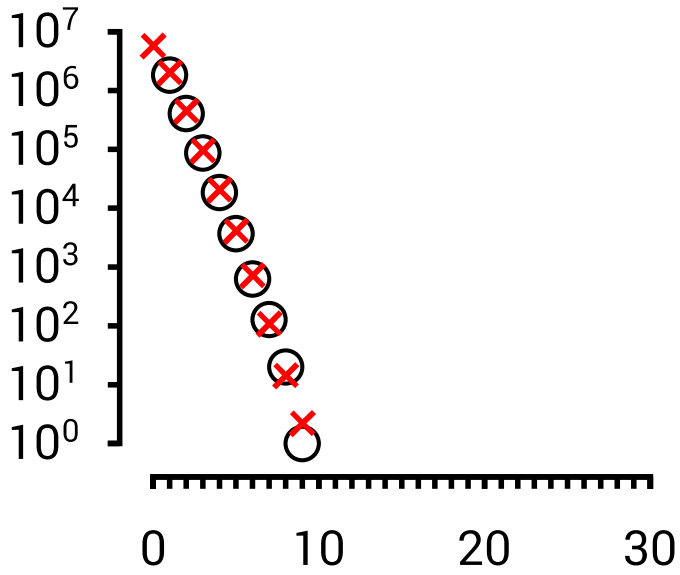
3.6.7. Example

```
python recon_v3.0.py -x --x_max 30 -o test_sample_1_plotfile.txt
-b error_bar_parameters.txt test_sample_1_fitfile.txt
```

Outputs a plotfile `test_sample_1_plotfile.txt` that contains the observed and refitted data side by side:

```
clone_size  sample    fit         lower_limit    upper_limit
0           0.0       5224621.0   3407038.40444  inf
1           1833459   1833117
2           405423    406083
3           86822     86315
4           18467     18638
5           3694      3722
6           626       651
7           128       99
8           20        13
9           1         2
```

as well as a PDF file with the same prefix as the input file, i.e., `test_sample_1_fitfile.pdf`:.



In this PDF, the x-axis is the clone size and the y-axis is the number of clones of that size. The black circles represent the observed data and red x's represent the resampled fit. "X mars the spot" represents a good fit. Note there is no circle for zero; the red x at zero is Recon's estimate of the number of missing species (i.e., the number of species that appear zero times in the sample).

3.7. Upper bound **`(-Umax, --upper_bound)`**

3.7.1. Description

Any reconstruction of missing species using a small sample from a large population suffers from a fundamental limitation. Species that are too rare to have an appreciable chance of appearing in the sample cannot be estimated based upon the sample. As shown by Mao and Lindsay, this results in upper confidence intervals for the missing species that are formally infinite.

As discussed, Recon addresses this problem by only estimating those species that are large enough to have an appreciable chance of influencing the sample distribution in a meaningful way. (Note that while mixing distributions are often approximated as continuous, in reality they are discrete, so smallest fitted population will often be practically meaningful.) In many cases this estimate will be of interest.

But this still leaves the estimate for all species unbounded. The number of individuals in a population is of course an upper bound for the number of species. In many cases of interest, such as analysis of immune repertoires, it is relatively easy to obtain reasonable estimates of the total number of individuals. For example, an estimate of total cells can be obtained by scaling a cell count against total tissue or blood volume, e.g., 10^{10} B cells in the body.

Below we show how the Recon fit can be combined with an estimate of the number of all individuals in a population to get a sharper upper bound on the number of all species.

Recon produces an overall clone-size distribution. The smallest clone size in this distribution is described by two parameters: the fraction of all clones that are of this size, wmin, and a mean

number of cells that it contributes to the sample, mmin. Clone sizes smaller than this contribute a mean of zero cells to the sample; however, it is possible that there are smaller clones in the parent population, clones so small that they both do not contribute to the sample and are invisible to our algorithm. Recon's estimate of the number of missing clones would not count such clones because it is not necessary to assume that they exist in order to obtain the observed sample clone-size distribution. However, if they were to exist, they would result in an undercount of the species richness in the parent. The goal in this section is to bound this potential undercount. One can then test its plausibility, as described in the main text.

The maximum undercount Umax, and therefore the desired upper bound, is obtained for the case that all the cells in clones smaller than mmin are actually singlets. How many would that be? The answer is given by

$$U_{max} = R \cdot w_{min} \cdot m_{min} \cdot N / S$$

where R is (Recon's upper bound of) the overall species richness estimate, N is the total number of cells in the overall repertoire, and S is the sample size. Note the ratio S/N is the fraction of cells in the overall population that are sampled; scaling m_{min} by S/N (yielding $m_{min}*N/S$) thus gives the smallest clone size in the overall repertoire that Recon can distinguish from singlets.

See Supplementary Information for Kaplinsky and Arnaout, *Nat. Comm.* 2016 for more details and examples (https://static-content.springer.com/esm/art%3A10.1038%2Fncomms11881/MediaObjects/41467_2016_BFncomms11881_MOESM1711_ESM.pdf).

3.7.2. Usage

```
python recon_v3.0.py -Umax -ff FIT_FILE -df D_NUMBER_FILE -N
population_size -ss sample_size
```

3.7.3. Required parameters

`-ff FIT_FILE`	A Recon fit file of the kind produced by running the –R option.
`-df D_NUMBER_FILE`	A Recon D-number file of the kind produced by running the –D option (which should contain output for the same input file for which the `FIT_FILE` is the Recon output)
`-N POPULATION_SIZE`	A float (no commas) of the size of the overall population from which the sample that went into producing the `FIT_FILE` was drawn
`-ss, --sample_size SAMPLE_SIZE`	A float (no commas) of the size of the sample that went into producing the `FIT_FILE`

3.8. Workflow

3.8.1. Typical case

Most commonly the goal will be to estimate the diversity (by the desired Hill number) of the overall populations from which a set of samples were drawn. For this, the workflow is to perform reconstruction on all samples (section 3.1); then, for each sample, plot a resampling from the model against the data to confirm goodness of fit (section 3.6); and finally to use pre-computed error bars to calculate diversity measures for all samples (section 3.3).

3.8.2. Custom error bars

If you want to explore diversity measures for *q*s other than the ones in the pre-computed error bars, you should first make a new error-bar parameter file (section 3.2) with the desired *q*s. If you want an error-bar profile based on your own custom set of distributions, first make those distributions; then perform reconstruction on them (section 3.1), placing the resulting fit files in their own directory; and then generate an error-bar paramter file based on those new fits (section 3.2).

3.8.3. Power tables

Power tables are most useful when planning experiments. Generating them is usually a standalone workflow that starts with the appropriate error-bar parameters (section 3.2) and involves just generating power tables for the desired *q*, total diversity, and sample size (section 3.4).

4. Contact information

Questions, comments, and other correspondence should be addressed to Ramy Arnaout at rarnaout@gmail.com.

5. License

5.1. Definitions

(a) “Derivative” means any translation, adaptation, alteration, transformation, or modification, including inclusion as part of another software program or product, of the Software.

(b) “Embedded Terms” means any terms and conditions of this Agreement embedded in the Source Code.

(c) “Intellectual Property Rights” means all patents, patent rights, patent applications, copyrights, copyright registrations, trade secrets, trademarks and service marks (including, where applicable, all derivative works of the foregoing).

(d) “Object Code” means computer programs assembled, compiled, or converted to magnetic or electronic binary form, which are readable and useable by computer equipment.

(e) “Software” means the software program known as, “Recon: Reconstruction of Estimated Communities from Observed Numbers”, including its Object Code and Source Code.

(f) “Source Code” means computer programs written in higher-level programming languages and readable by humans.

5.2. License

Subject to the terms and conditions of this Agreement, BIDMC grants to Licensee a no cost, personal, non-exclusive, non-transferable, limited license (without the right to sublicense) to download the Software, and to copy, make Derivatives and use the Software for Licensee’s internal academic and research purposes during the term of this Agreement. Any rights not expressly granted in this Agreement are expressly reserved.

(a) Derivatives. Licensee agrees that from time to time, or upon request by BIDMC, Licensee will deliver all Derivatives to BIDMC and hereby grants to BIDMC a no cost, personal, perpetual, irrevocable, non-exclusive, non-transferable, limited license (with the right to sublicense) to download the Derivatives, to copy, distribute and make Derivatives of the Derivative, and to use the Derivative for BIDMC’s internal academic and research purposes.

(b) Commercial Restrictions on Use of the Software. The following are prohibited without obtaining a commercial license from the Office of Technology Ventures at BIDMC:

(i) Using the Software or any Derivative to produce any commercial product or to provide any commercial service.

(ii) Charging a fee for use of the Software or any Derivative for any purpose.

(iii) Distributing the Software or any Derivative to any other party, unless the distribution is made subject to this Agreement and the recipient “Agrees” to this Agreement through the website:

https://github.com/ArnaoutLab/Recon

(c) Patents. BIDMC does not grant through this Agreement any licenses under any BIDMC patent or patent application.

(d) Intellectual Property Rights Notices; Embedded Terms. Licensee is prohibited from removing or altering any of the Intellectual Property Rights notice(s) and any Embedded Terms embedded in the Software. Licensee must reproduce the unaltered Intellectual Property Rights notice(s) and the Embedded Terms in any full or partial copies of the Source Code that Licensee makes.

(e) Export Compliance. Licensee acknowledges and agrees that U.S. export control laws and other applicable export and import laws govern download and use of the Software. Licensee will neither export nor re-export, directly or indirectly, the Software in violation of U.S. laws or use the Software for any purpose prohibited by U.S. laws.

5.3. Disclaimer of Warranties; Limited Liability

(a) Disclaimer of Warranties. THE SOFTWARE IS DELIVERED "AS IS." BIDMC MAKES NO OTHER WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED, WITH REGARD TO THE SOFTWARE PROVIDED UNDER THIS AGREEMENT, IN WHOLE OR IN PART. BIDMC EXPLICITLY DISCLAIMS ALL WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY AND OF FITNESS FOR A PARTICULAR PURPOSE. BIDMC EXPRESSLY DOES NOT WARRANT THAT THE SOFTWARE, IN WHOLE OR IN PART, WILL BE ERROR FREE, OPERATE WITHOUT INTERRUPTION OR MEET LICENSEE'S REQUIREMENTS.

(b) Limited Liability; No Consequential Damages. THE TOTAL LIABILITY OF BIDMC, ITS AFFILIATES, TRUSTEES, OFFICERS, AND EMPLOYEES IN CONNECTION WITH THE SOFTWARE, OR ANY OTHER MATTER RELATING TO THIS AGREEMENT (WHATEVER THE BASIS FOR THE CAUSE OF ACTION) WILL NOT EXCEED IN THE AGGREGATE OVER THE TERM OF THE AGREEMENT 00. IN NO EVENT WILL BIDMC , ITS AFFILIATES, TRUSTEES, OFFICERS, AND EMPLOYEES BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR DAMAGES FOR LOST PROFITS, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY, OR OTHERWISE.

(c) Failure of Essential Purpose. THE LIMITATIONS SPECIFIED IN THIS SECTION WILL SURVIVE AND APPLY EVEN IF ANY REMEDY SPECIFIED IN THIS AGREEMENT IS FOUND TO HAVE FAILED OF ITS ESSENTIAL PURPOSE.

5.4. Termination

(a) Right of Termination. BIDMC may, for any reason or no reason, upon written notice sent to the contact information Licensee provides upon clicking "agree", immediately terminate this Agreement. This Agreement will automatically terminate upon any breach by Licensee of any of the terms or conditions of this Agreement. BIDMC shall not be liable to Licensee or any third party for any termination of this Agreement.

(b) Licensee Derivatives. Upon termination or expiration of this Agreement, Licensee shall promptly deliver to BIDMC the Source Code and Object Code of Licensee's Derivatives and shall immediately cease all use of the Software.

5.5. Non-use of Name

Without BIDMC's prior written consent, Licensee will not identify BIDMC in any promotional statement, or otherwise use the name of any BIDMC employee or any trademark, service mark, trade name, or symbol of BIDMC.

5.6. Assignment

This Agreement and the rights and obligations hereunder are personal to Licensee, and may not be assigned or otherwise transferred, in whole or in part, without BIDMC's prior written consent. Any attempt to do otherwise shall be void and of no effect. BIDMC has the right to assign this Agreement or any rights or obligations hereunder to any third party. This Agreement shall be binding upon, and inure to the benefit of, the successors, representatives and permitted assigns of the parties.

5.7. Choice of Law

This Agreement and all disputes and controversies related to this Agreement, are governed by and construed under the laws of the Commonwealth of Massachusetts, without regard to the choice of law provisions. The state and federal courts located in the Commonwealth of Massachusetts are the exclusive forum for any action between the parties relating to this Agreement. Licensee submits to the jurisdiction of such courts, and waives any claim that such a court lacks jurisdiction over Licensee or constitutes an inconvenient or improper forum. The United Nations Convention on the International Sale of Goods (CISG) shall not apply to the interpretation or enforcement of this Agreement.

5.8. English Language

This Agreement is originally written in the English language and the English language version shall control over any translations.

5.9. Entire Agreement

This Agreement constitutes the entire agreement, and supersedes all prior and contemporaneous agreements or understandings (oral or written), between the parties about the subject matter of this Agreement. Licensee has no right to waive or modify any of this Agreement without the written consent of BIDMC. No waiver, consent or modification of this Agreement shall bind either party unless in writing and signed by the party granting the waiver. The failure of either party to enforce its rights under this Agreement at any time for any period will not be construed as a waiver of such rights. If any provision of this Agreement is determined to be illegal or unenforceable, that provision will be limited or eliminated to the minimum extent necessary so that this Agreement will otherwise remain in full force and effect and enforceable.

Last Updated February 8, 2016