

SQEMA manual - version 0.5

Simon Jarman, February 2025

Contents

1. Introduction
 - 1.1 Data
 - 1.2 Installation
 - 1.3 Control
 - 1.4 Help
 - 1.5 File input
 - 1.6 Sqema output
 - 1.7 Sqema code
2. Sqema functions
 - 2.1 Fitting SAD models
 - 2.2 Estimating relative species abundance
 - 2.3 OTU table column/sample modification
 - 2.4 OTU table row/OTU modification
 - 2.5 Simulation
 - 2.6 Community comparisons
 - 2.7 Dispersion metrics
 - 2.8 Biodiversity metrics
3. Example workflow
4. Alphabetical command list
5. References

1. Introduction

Sqema (Species Quantification from Environmental nucleic acid Metabarcoding Abundance) is a software package for generating species abundance distributions from data taken from environmental DNA or RNA (eNA) data sets.

SQEMA works on the assumption that the abundance of species in environments that eNA is sampled from follow predictable patterns of abundance. These "species abundance distributions" (SADs) have been studied extensively by ecologists and they provide valuable null hypotheses for the real abundance of species.

1.1 Data

Sqema works from nucleic acid read counts in Operational Taxonomic Unit (OTU) "OTU tables" formatted as .csv files. The term "OTU table" means in this context any table of nucleic acid sequence read counts derived from metabarcoding where the reads are aggregated to a species level. ZOTU (Zero-radius OTU) and ASV (Amplified Sequence Variant) tables can have reads for individual sequence variants within a species aggregated to species level to fit the assumptions that sqema is based on. The software has functions to help with this where names can be ascribed to variants, or it can be done by clustering in other software such as USEARCH (Edgar 2010) or VSEARCH (Rognes et al. 2016).

The input file format must conform to having one title row with the names of the samples in it and one title column with the names of the OTUs in it. The rest of the cells in the table contain DNA or RNA sequence read counts. These should be raw counts as an initial input, not proportionalised normalised, or rarefied. An example of this format is:

	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
OTU a	45	23	0	55	87
OTU b	690	789	0	1123	45
OTU c	0	0	0	0	0
OTU d	234	123	0	2	88
OTU e	98	554	0	12	1188

Sqema has commands for OTU table editing that help convert complex OTU tables into this format (sections 2.3 and 2.4)

Sqema will remove any rows or columns that have only zeros in them before analysis, so in this example, both Sample 3 and OTU c would be removed from the table before analysis. The outputs will not include OTU c or sample 3. If sample or OTU names are duplicated, sqema will raise an error for any of the quantitative analyses telling the user what the duplicates are. These must be removed and the file reloaded, either with sqema OTU table manipulation commands, or by editing in a spreadsheet or similar software package.

1.2 Installation

Sqema can be installed using the Python package installer. To do this, open a terminal on Linux/OSX systems and type:

```
python3 -m pip install --upgrade pip
```

This ensures that an up-to-date version of pip is installed. Then type:

```
pip install sqema --target ~/sqema_all/
```

This will install sqema to the home folder on OSX contained in a folder called 'sqema_all' (you can call this what you like).

To run sqema on Windows, the recommended approach is to use a Linux distribution in a virtual machine. There are a few options for this, but I have had success with VirtualBox:

<https://www.virtualbox.org/>

1.3 Control

To use sqema, first navigate to the folder containing the script `sqema.py` with something like this:

```
cd Users/273915i/sqema_all/sqema/
```

... which on my system is the folder where I have installed sqema.py. On Linux systems, try:

```
cd /Home/sqema_all/sqema/
```

or something similar. **Once you have navigated to the folder containing sqema.py, then all other commands in this manual, which begin `python sqema.py` should work.**

This is not the only way to run it, and it should be possible to change permissions and PATH variables etc. so that sqema.py is executable and you don't have to enter "python" before it, but this will be platform-specific. For ease of instruction, in this manual, I will stick to the simplest and most robust option. Note that different versions of Python may be installed on your system. If so, it may be desirable to use `python3`, or `python3.11` instead of `python`, as needed.

Sqema is run from a command line interface. The user controls the base function that sqema will run for each operation by specifying a function and a set of parameters for the function. The base functions for SQEMA have to be specified immediately after the `sqema` command to run the program. Parameters for the

function are specified either in long form with a '-' before the verbose parameter name, or preceded by a '-' symbol for the abbreviated name for the parameter. For example, '-in' and '--in_file' do the same thing, but the first needs one hyphen before it, and the second two.

Each part of the overall command input must be separated by a space, except in parameter values which are lists, where commas are used. An example is:

```
python sqema.py fit_SAD -in sqematest_1.csv
```

An example with input in a list is:

```
python sqema.py fit_SAD -tmo zipf logser lognorm
```

where the --test_models parameter is given with its short-form -tmo as a list of three models to test. Items in a list should be entered separated by spaces. If the name has a space in it, put it in quotes, e.g. "Genus species".

Each parameter has a specified type:

"Str" is a string of characters, such as a name of a sample.

"Choice" requires selection of one of a list of strings

"Choices" requires selection of one or more strings entered as a list with spaces between the items, e.g.

"float" is a floating point decimal number like 0.993

"int" is an integer number with no decimals like 3

All of the parameters have default values that are used if a user-defined value is not specified. For example, the default for -in / --in_file is sqematest_1.csv. This means that if the user does not specify an input file, sqema will analyse the sqematest_1.csv file that is included with the package, for example with this command:

```
python sqema.py fit_SAD
```

which would run the SAD fitting algorithm on the default data, and also use default parameters for the variables.

1.4 Help

The full list of available commands in both abbreviated and verbose form can be displayed with the command

```
python sqema.py -h
```

1.5 File input

Sqema takes input from .csv formatted files. It will search in the same directory as the script is located, so:

```
python sqema.py fit_SAD -in sqematest_1.csv
```

Means that it will look in the same directory where sqema is installed for the file sqematest1.csv to use as its input. If you want to use files in a different location, specify this in the standard system format, for example:

```
python sqema.py fit_SAD -in ~/Dropbox/sqema_inputs/test_1.csv
```

sqema does some automated data cleanup on OTU tables. It removes samples (columns), and taxa (OTUs) that only contain zeros. An OTU table that has been filtered to remove either samples or taxa may end up with either of these issues, but these zeros are not useful in sqema's functions, so it removes them and reports this to the standard output. The default test file has one column of zeroes in it to demonstrate this, and the removal will be reported to the terminal like this when you load the default file:

```
Zero only columns removed: 100291,
```

If a persistent record of standard output is required, on Linux/OSX, the output can be sent to a textfile to make a record like this:

```
python sqema.py fit_SAD -in  
~/Dropbox/sqema_inputs/sqematest_1.csv > ~/Desktop/sqema_log.txt
```

Lists of parameters should be entered separated by spaces. If the items being entered have a space in them, they should be entered with quotes around that item. For example:

```
python sqema.py rm_rows -in ~/Desktop/Krill_eDNA.csv -b names -  
rm_names Thysanoessa 'Euphausia superba'
```

would remove OTUs named Thysanoessa or Euphausia superba from the table.

1.6 sqema output

Output from sqema is always to a directory. The default directory is 'sqema_output' in the same directory as the main sqema scripts. To direct output to a different directory, use the **-od** command, like this:

```
python sqema.py fit_SAD -in
~/Dropbox/sqema_inputs/sqematest_1.csv -od
~/Desktop/custom_sqema_output_directory/
```

By default, sqema will rename output files based on the input file. A timestamp is added and a description of what was done to make the new file. For example:

```
python sqema.py fit_SAD -in sqematest_1.csv
```

will produce file in the sqema_output directory that have times associated with them, and will report this in the command linewith something like this:

```
File saved: sqema_output/sqematest_1_Thu, 30 Jan 2025 16-24-
25_fit_SAD.png
```

```
File saved: sqema_output/sqematest_1_Thu, 30 Jan 2025 16-24-
25_fit_SAD.csv
```

If a definitive filename and location are required, use the **-od** command for the directory, and the **-of** command for the desired filename.

For example,

```
python sqema.py bdiv_metrics
```

Will analyse the default sqematest_1.csv file and write the result to the default sqema_output/ directory, as reported:

```
File saved: sqema_output/sqematest_1_Thu, 30 Jan 2025 16-25-
16_bdiv_metrics.csv
```

If you do it again shortly afterwards, a new file with a different timestamp is written:

```
File saved: sqema_output/sqematest_1_Thu, 30 Jan 2025 16-26-
19_bdiv_metrics.csv
```

This behaviour is useful if you want to test sqema functions, and not have files overwrite each other.

If you want to send the output to another directory, use `-od` like this:

```
python sqema.py bdiv_metrics -od ~/Desktop/sqema_biodiv_results/
```

Which will save a folder on the desktop called "sqema_biodiv_results" with a file in it with the default name of "biodiversity_metrics.csv":

```
File saved:
/Users/273915i/Desktop/sqema_biodiv_results/sqematest_1_Thu, 30
Jan 2025 16-26-59_bdiv_metrics.csv
```

If you also want a specified name, for instance if you are feeding the output of sqema into another command, use the `-of` command:

```
python sqema.py bdiv_metrics -od ~/Desktop/sqema_biodiv_results/
-of BD_table_1
```

which will give the same folder with the results file called BD_table_1.csv in it. Note that the file extension is added automatically by sqema.

1.7 sqema code

Sqema is written in Python 3 and requires the following packages:

```
scipy, version >=1.11
numpy, version >=1.26
matplotlib version >= 3.8
pandas version >=2.1
svg version >=1.5.0
```

The source code is editable, of course, where default parameters are not as desired, or if new functions are to be included. The license under which the software is released allows modification and redistribution if credit for the original code is given.

2. Sqema functions

2.1 Fitting SAD models

fit_SAD finds SAD models that are the best fit to eDNA metabarcoding read abundance data. Each fit of a model to the data requires:

1. A specified *transform* of the data, if any.
2. A *metric* for measuring the read abundance found in the dataset.
3. An SAD *model*, to which SQEMA fits the parameters of so that the best fit to the data can be found,

Fitting will fit multiple combinations of transform, metric, and model simultaneously and report how well each combination fits based on likelihood and the Akaike Information Criterion (Akaike 1974).

The transforms implemented in sqema are:

untransf is the untransformed data from the OTU table. This is the default value and means that the read counts are unmodified.

arcsin_sqrt is an arcsine square root transform (https://en.wikipedia.org/wiki/Arcsine_distribution).

expit_lim is an approximation of the expit transform (the inverse of a logit transform). Artificial limits set to avoid the requirement for an infinite distribution (https://en.wikipedia.org/wiki/Logistic_function).

clr is the "central log ratio" transform. This transform is recommended for compositional data like eDNA metabarcoding results (Aitchison 1983; Gloor et al. 2017; Quinn et al. 2018).

The metric implemented in sqema are:

RRA is "relative read abundance" following the definition in [\(Deagle et al. 2019\)](#). This is the proportionalised read counts per sample.

RwRRA is "R weighted RRA". In this metric, a weighting applied to samples based on the species diversity (R) present in them. Samples with higher R are likely to have eDNA that provides a more comprehensive representation of the true biodiversity present in the environment. The weighting is implemented as:

RRA x R/maxR

POO is “proportion of occurrence” as given in [\(Deagle et al. 2019\)](#). This is the floating point equivalent of “frequency of occurrence” which is often given as a percentage, so 50% FOO is 0.5 POO.

wPOO is “weighted proportion of occurrence” as given in [\(Deagle et al. 2019\)](#).

The models implemented in sqema are:

logser for the log-series model
(https://en.wikipedia.org/wiki/Logarithmic_distribution) of SAD (Fisher, Corbet, and Williams 1943).

lognorm for the lognormal distribution (https://en.wikipedia.org/wiki/Log-normal_distribution) as a SAD (Preston 1948).

geom for the geometric distribution
(https://en.wikipedia.org/wiki/Geometric_distribution) used as a SAD.

zipf for the Zipf distribution (https://en.wikipedia.org/wiki/Zipf%27s_law) (Zipf 1935) used as a SAD.

genpareto for the generalised Pareto distribution
(https://en.wikipedia.org/wiki/Generalized_Pareto_distribution) for a SAD model.

nbinom for the negative binomial distribution
(https://en.wikipedia.org/wiki/Negative_binomial_distribution) (Rémond de Montmort, n.d.) used for a SAD.

linear for a 1:1 relationship between the metric and biomass. Useful for the purposes of comparing un-fitted data with data fitted to the other models.

Parameters:

-in -in_file

Type = str, default = 'test_1.csv'

-out -out_dir

Type = str, default='sqema_output'

--ttr --test_transforms

Type = choices = 'untransf','arcsin_sqrt','expit_lim', 'clr', default = 'untransf'

Multiple choices as a comma separated list without spaces.

Transforms are applied to the entire dataset before other calculations.

-tme --test_metrics

Type = choices=['RRA','RwRRA','POO','wPOO','PRD','PRA'],default='RRA'
Metrics are assigned to each OTU in a sample based on calculations derived from read count abundance. Multiple choices as a comma separated list without spaces.

-tmo --test_models

Type = choices= 'logser','zipf','genpareto','lognorm','geom','linear','nbinom',default='logser'. Multiple choices as a comma separated list without spaces.

A range of SAD models to test for fit to metabarcoding data.

-gf --graph_format

Type = choices=['.svg','.png','.jpg'],default='.png'

Format for plotting the graphs. .svg for vector graphics, allowing editing in Inkscape (<https://inkscape.org/>) or similar software, .png and .jpg for raster files.

-py --plot_yaxis

Type = choices=['log','linear'],default='log',help=''

The y-axis can be set to either a log or linear scale.

The outputs of model fitting are a graphical representation of the best fitting models and a .csv file of the relevant data.

Example

```
python sqema.py fit_SAD -in sqematest_1.csv -tmo logser lognorm  
nbinom zipf -tme RRA
```

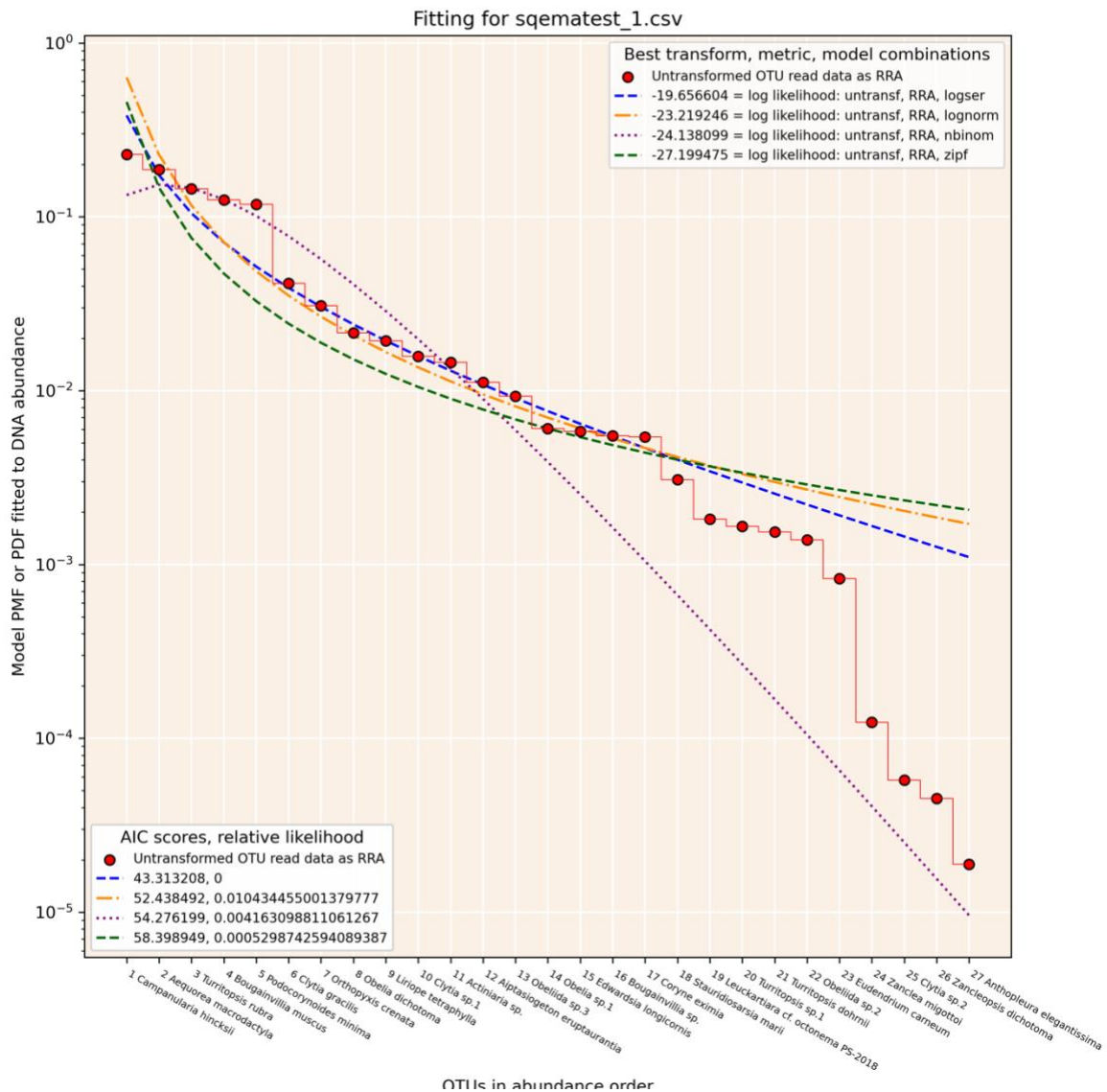
This will fit four different SAD models to the default dataset `sqematest_1.csv`.
The data is

Result:

This fit of the four specified SAD models is given in a .csv file as a table like this:

Log likelihood	Transform	Metric	Model	Model parameters	AIC	Relative Likelihood
-19.656604	untransf	RRA	logser	p: 0.9064239977079377; loc: 0.0; s: 1.578050229179029; loc: 0.9; scale:	43.3132076	0
-23.219246	untransf	RRA	lognorm	1.3817664886790468;	52.4384916	0.01043446
-24.138099	untransf	RRA	nbinom	p: 3.0; n: 0.42633377435532227; loc: 0.0;	54.2761988	0.0041631
-27.199475	untransf	RRA	zipf	a: 1.6391426973287038; loc: 0.0;	58.3989493	0.00052987

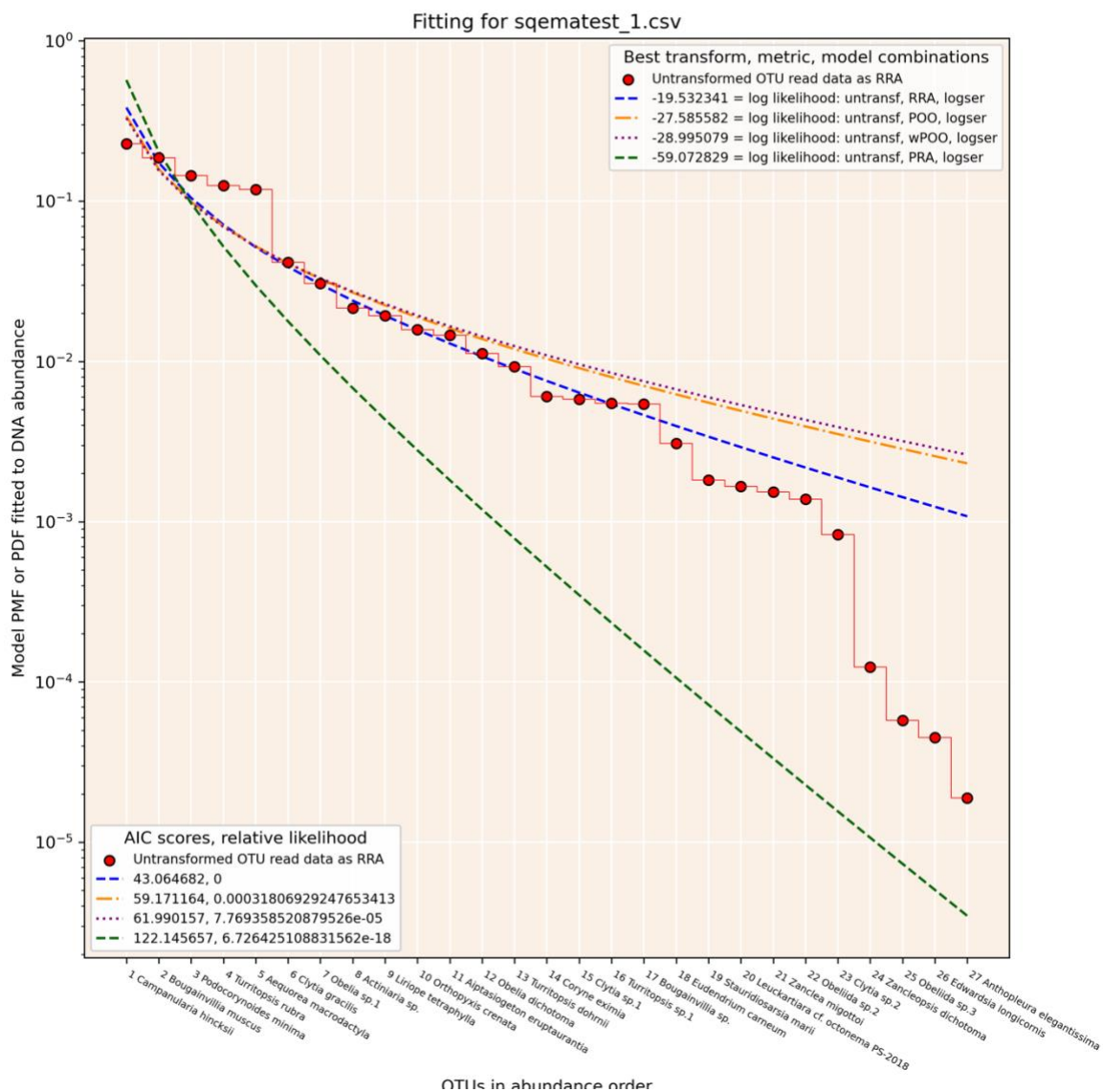
This shows the likelihood of the models and AIC scores. The best-fitting model in this case is the logseries model. There is also a graphical representation of the fit shown below, which makes the logser fit appear the best option as well.



Having determined that the logser is the best model, we could see if a different metric would improve the fit with a command like this

```
sqema fit_SAD -in sqematest_1.csv -tmo logser -tme RRA PRA POO
wPOO
```

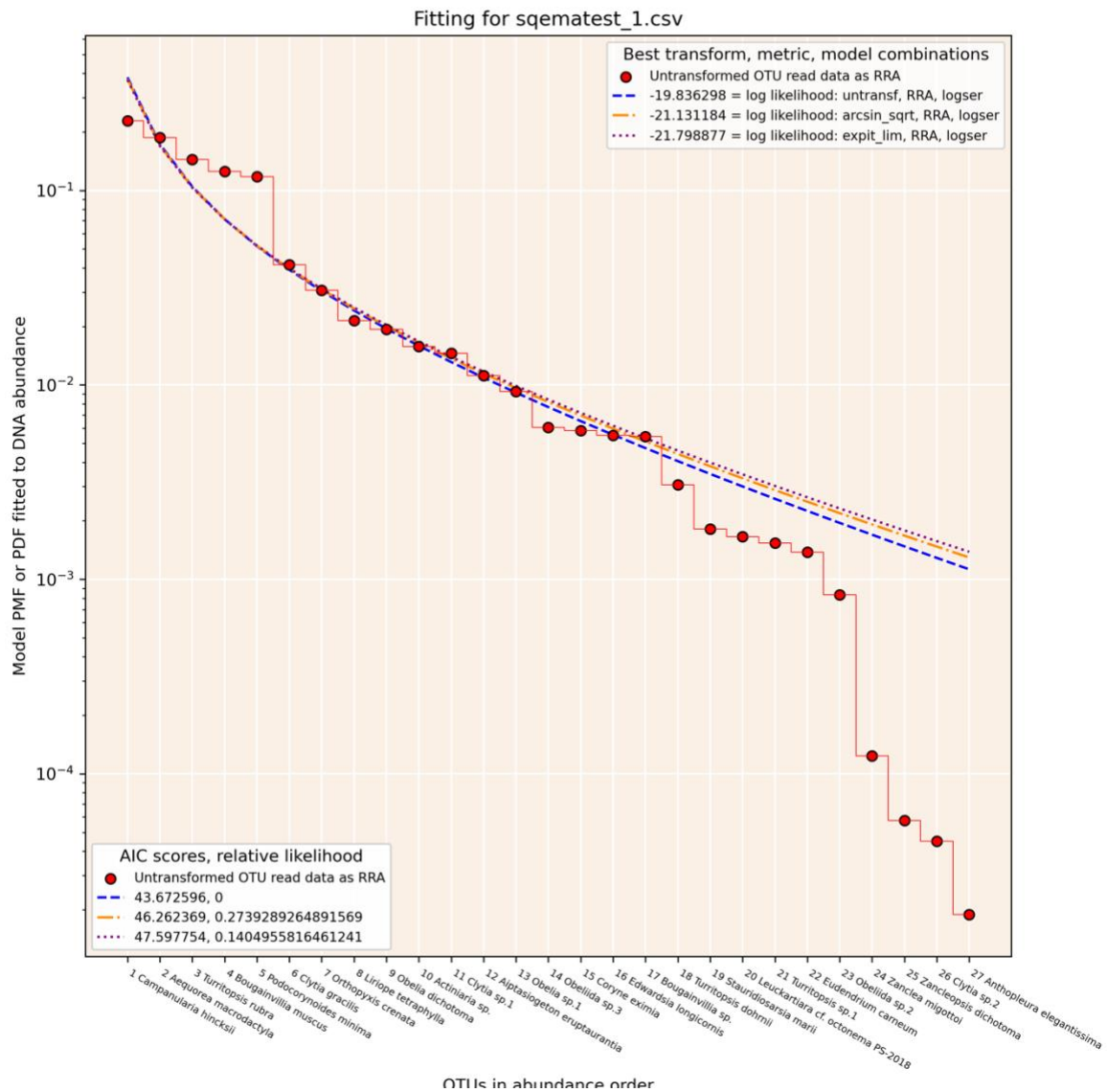
Just looking at the graphical output, it seems that RRA is the best option:



With a logser model and RRA, we could check whether any data transforms are beneficial for the fit with this:

```
python sqema.py fit_SAD -in sqematest_1.csv -tmo logser -tme RRA
-ttr untransf arcsin_sqrt expit_lim
```

The graph below shows that the untransformed data is a slightly better fit than expit or the arcsinsqrt transformed data, so it looks like the logseries model with RRA as the metric and untransformed data is the best option for quantification.



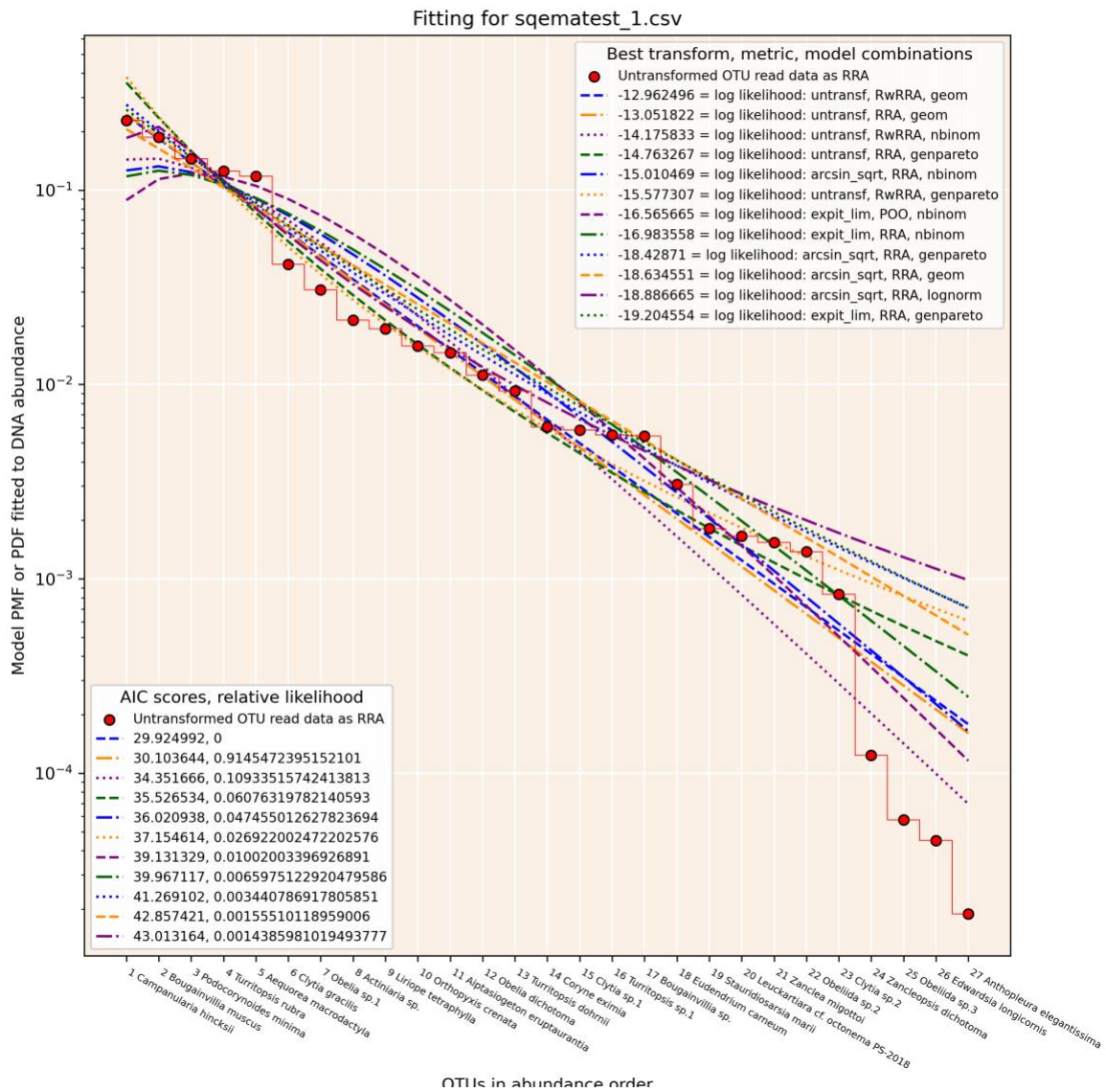
However, it is possible to check all implemented combinations of model, metric, and transform. This command:

```
python sqema.py fit_SAD -tmo logser lognorm nbinom zipf
genpareto geom -tme RRA RwRRA PRA PRD POO wPOO -ttr untransf
arcsin_sqrt expit_lim clr
```

will test the default sqematest_1.csv file for all combinations of these. This analysis shows that it is possible to find a better-fitting model, with the tabulated results showing that the geometric model with R-weighted relative read abundance and no data transforms is the closest fit:

Log likelihood	Transform	Metric	Model	Model parameters	AIC	Relative Likelihood
12.962496	untransf	RwRRA	geom	p: 0.2422414423551688; loc: 0.0;	29.9249916	0
13.051822	untransf	RRA	geom	p: 0.245742631346948; loc: 0.0;	30.1036439	0.91454724
14.175833	untransf	RwRRA	nbinom	p: 2.0; n: 0.3263703311000005; loc: 0.0;	34.3516661	0.10933516
14.763267	untransf	RRA	genpareto	c: 0.15724241630382177; loc: 0.9; scale: 2.683066415350518;	35.5265335	0.0607632
15.010469	arcsin_sqrt	RRA	nbinom	p: 2.0; n: 0.3002011987366431; loc: 0.0;	36.0209378	0.04745501
15.577307	untransf	RwRRA	genpareto	c: 0.23965940562780455; loc: 0.9; scale: 2.4951633077029;	37.1546144	0.026922
16.565665	expit_lim	POO	nbinom	p: 3.0; n: 0.3586392096453717; loc: 0.0;	39.1313292	0.01002003
16.983558	expit_lim	RRA	nbinom	p: 2.0; n: 0.28728813284136184; loc: 0.0;	39.9671168	0.00659751
-18.42871	arcsin_sqrt	RRA	genpareto	c: 0.11315313454467357; loc: 0.9; scale: 3.5228406616198225;	41.2691017	0.00344079
18.634551	arcsin_sqrt	RRA	geom	p: 0.2057931015146118; loc: 0.0; s: 0.9326284037893275; loc: 0.3036024447730369; scale: 2.950290452003355;	42.8574209	0.0015551
18.886665	arcsin_sqrt	RRA	lognorm	c: 0.0849517163793254; loc: 0.9; scale: 3.7468620195765223;	43.0131639	0.0014386
19.204554	expit_lim	RRA	genpareto		43.5931734	0.00107645
19.506582	untransf	RRA	logser	p: 0.905518099716806; loc: 0.0;	43.7733304	0.00098372
19.796587	untransf	RwRRA	logser	p: 0.9071150408653158; loc: 0.0;	44.4091075	0.00071584

And the graphical results are:



... although they are hard to interpret with so many curves on one graph. The effect of R-weighting RRA scores is very minor, so there could be an argument for choosing RRA alone as reducing the number of data manipulations may be wise. Ultimately, it is up to the user to decide and justify their choices.

2.2 Estimating relative species abundance

quantify uses an SAD model with parameters estimated from an OTU table to estimate the relative abundance of the OTUs in the table. This takes SAD model parameters derived from the fitting command (section 2.1) or determined by other means. The outputs of model fitting are two graphical representation of the best fitting models and a .csv file including all of the relevant data.

-in -in_file
Type = str, default = 'test_1.csv'

-out -out_dir
Type = str, default='sqema_output'

-tr -transform
Type = choices = 'untransf','arcsin_sqrt','expit_lim', 'clr', default = 'untransf'
A single choice is required.
Transforms are applied to the entire dataset before other calculations.

-me --metric,
Type = choices=['RRA','RwRRA','POO','wPOO','PRD','PRA'],default='RRA'
Metrics are assigned to each OTU in a sample based on calculations derived from read count abundance. A single choice is required.

-mo -model
Type = choices= 'logser','zipf','genpareto','lognorm','geom','linear','nbinom',default='logser'. A single choice is required.
The SAD models relate the rank abundance of species to their real abundance.

-pr --pseudoreplicates
Type = int, default = 500
The number of pseudoreplicate datasets generated for estimating confidence in rank differences.

-mfp --max_fit_plot
Type = int, default = 12
Maximum number of the best fitted SADs to plot

-p --p_value
Type=float, default=0.05
The p value for Brunner-Munzel tests for rank differences assessed on pseudoreplicated data.

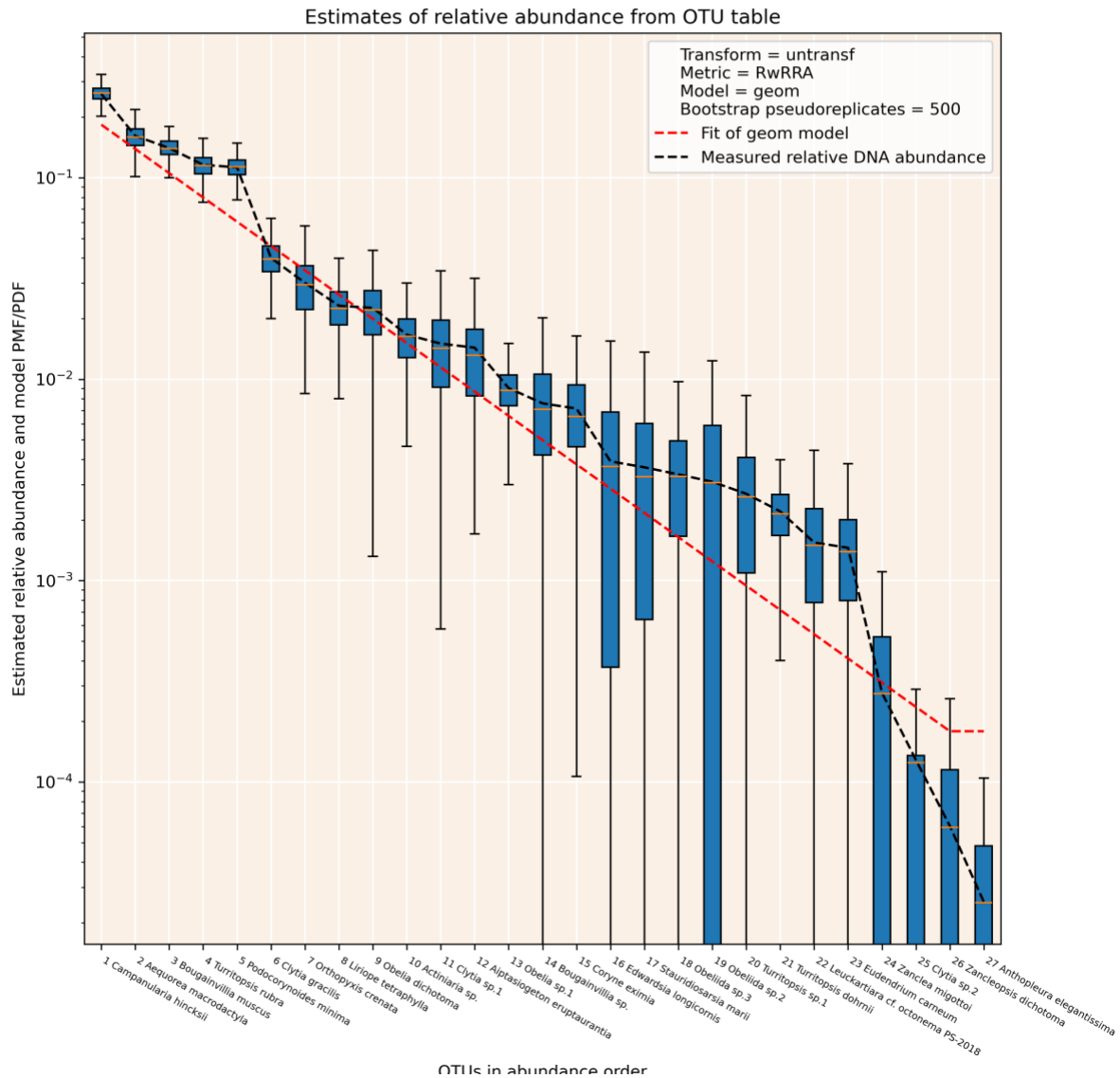
Example:

Using the parameters that the **fit_SAD** command suggested are the best fit to the default sqematest_1.csv data, we use this command:

```
python sqema.py quantify --model geom --metric RwRRA
```

Result:

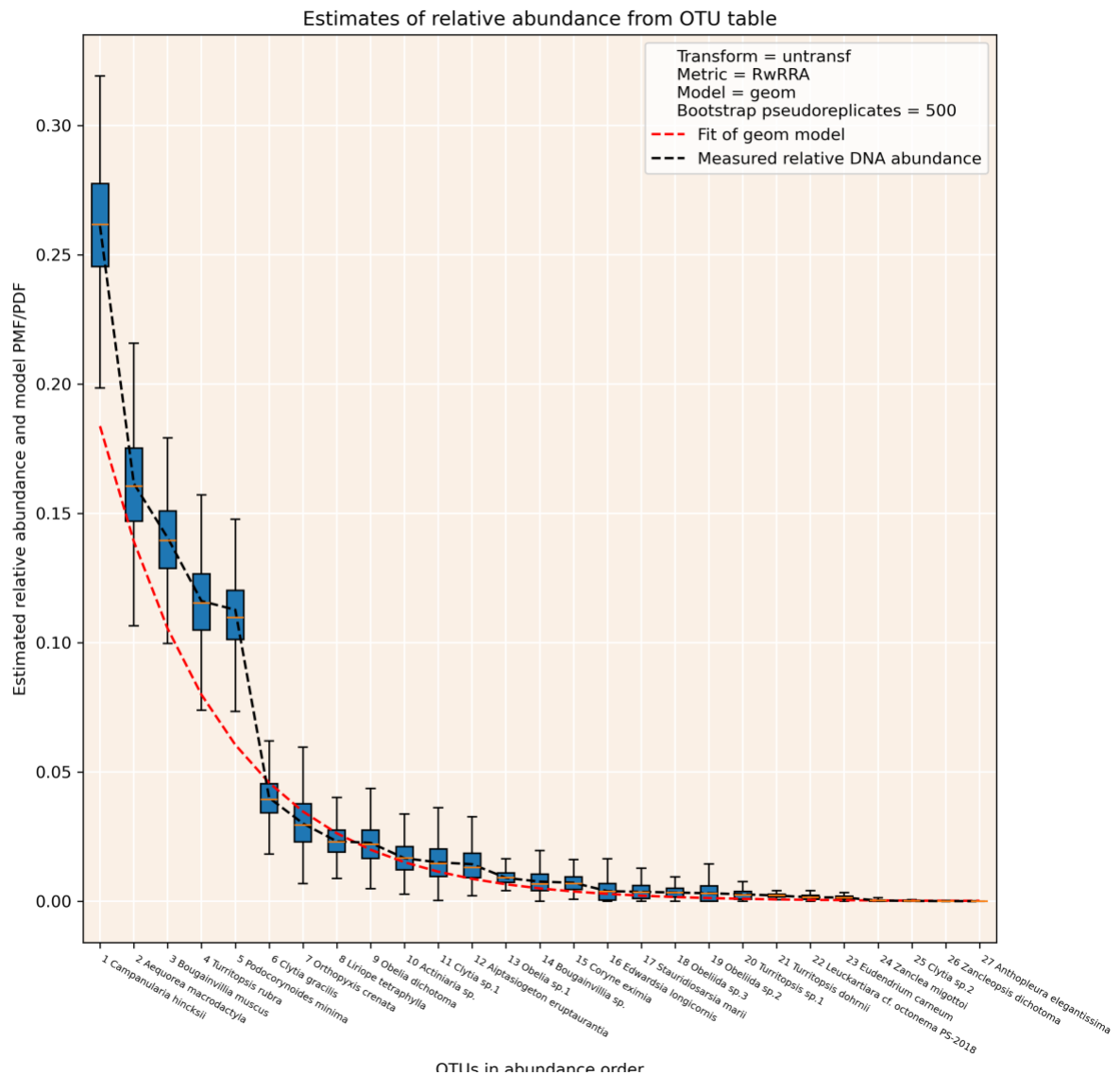
This gives two graphs and one .csv table as output. The first graph shows the results of bootstrap resampling the data. The range of estimated relative abundance values is shown as a box-plot.



The y axis is a log scale by default, so the range of values appears larger for the low abundance species. To see the same result on a linear plot, use the -py linear command like this

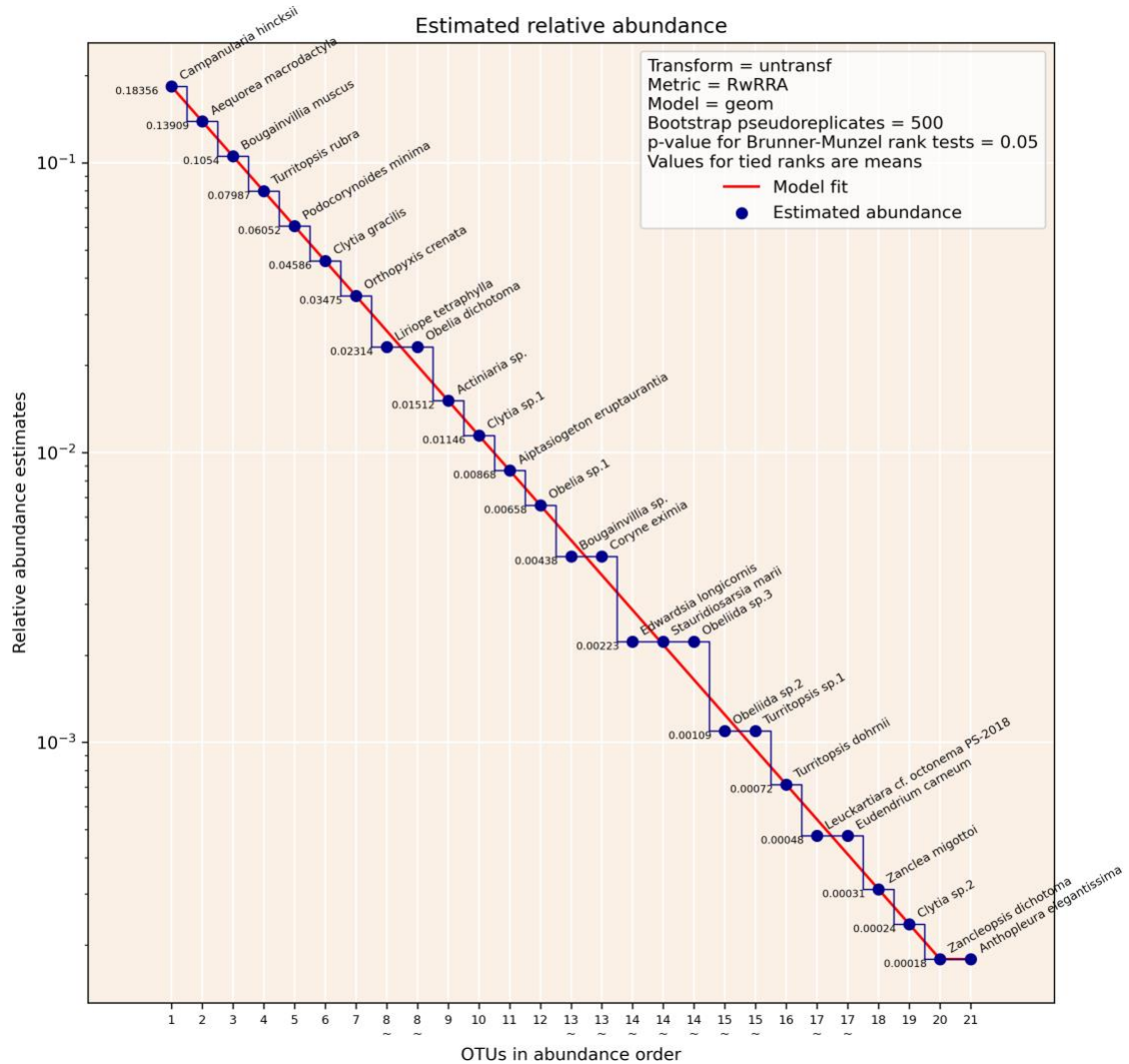
```
python sqema.py quantify --model geom --metric RWRRA -py linear
```

which will give:

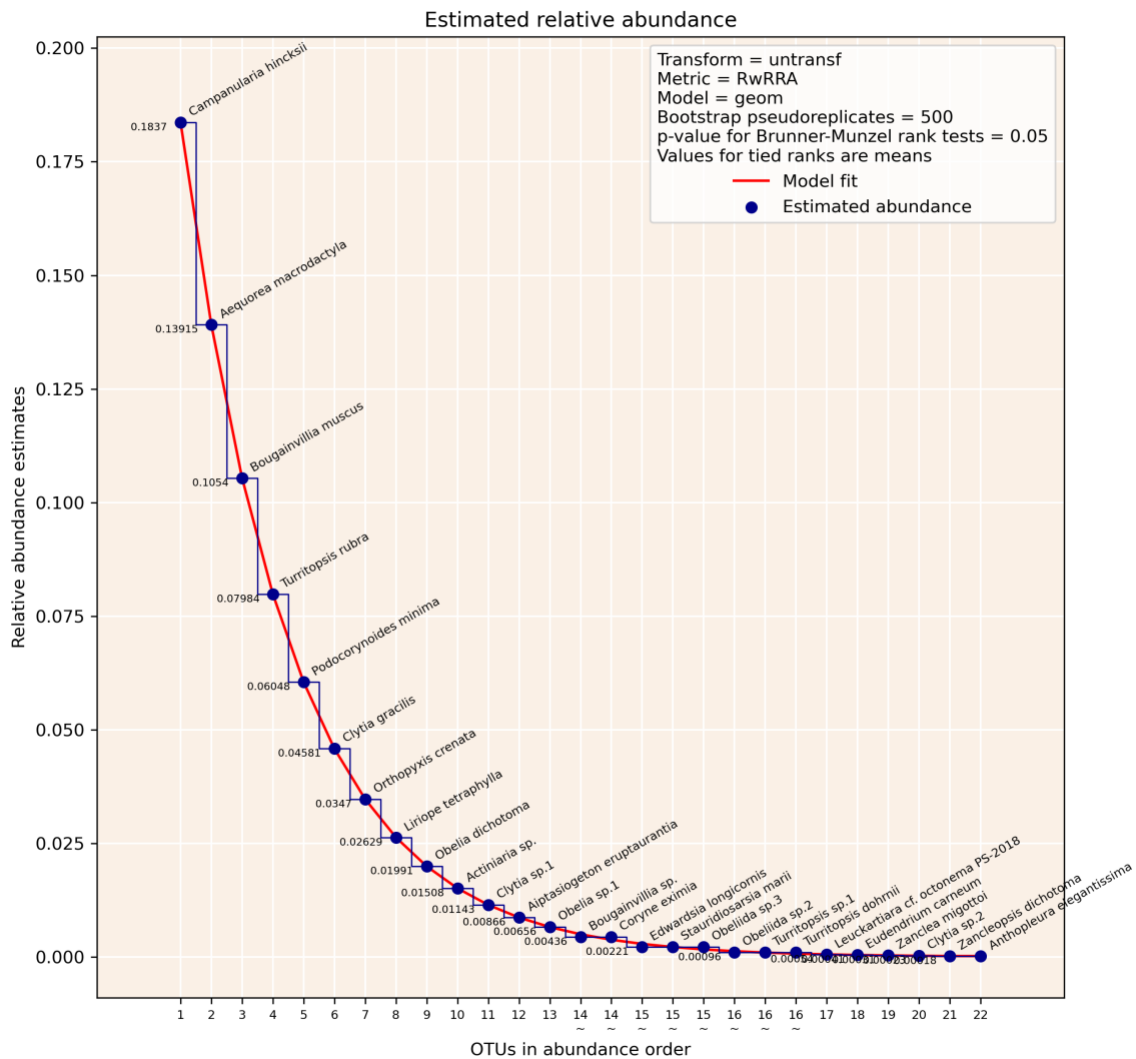


Which can be easier to interpret. The range of bootstrap values is used to decide whether species/OTUs have genuinely different abundance ranks. A Brunel-Munzer test is used to determine whether there is a significant difference in the ranges of values found in the pseudoreplicated data. If there is, then ranks are assigned based on the mean values. If not, ranks are tied and several OTUs can have a shared rank.

The graphical result of this with a log axis is useful for seeing the ranks in the lower abundance species. Where a rank is tied, a tilde indicates this on the x-axis.



The same graph with a linear y-axis looks like this:



Which makes it easier to imagine real proportional species abundances. As is commonly-seen, most of the OTUs are low in abundance and a few OTUs dominate the community.

2.3 OTU table column/sample modification

rm_columns uses rules to remove samples/columns from an OTU table. The OTU table is then written to a new .csv file.

-b --basis

Type = choices= names, low_reads, richness, default = 'names'.

Defines the basis for removing samples, either by names, read counts, or OTU richness.

-rn --rm_names

Type = str, default = "

If "names" is specified for **-basis**, the names of samples (columns) to remove from an OTU table.

-minR --minimum_R

Type = int, default = 2

Minimum OTU richness for samples in an OTU table to be retained.

-minReads --minimum_reads

Type = int, default = 5

Minimum number of reads for samples in an OTU table to be retained.

Example_1:

To remove all samples in the OTU table with fewer than 100 reads from the default dataset, this is the command:

```
python sqema.py rm_columns -b low_reads -minReads 100
```

Result_1:

This removes all of the samples containing less than 100 reads and reports which ones:

Low read samples removed:

```
100198,100210,100213,100219,100234,100257,100269,100270,100272,100277,100279,100282,100285,100287,100290,100291,100297,100304,100309,
```

Example_2:

To remove all samples in the OTU table that have only one OTU, do this:

```
python sqema.py rm_columns -b richness -minR 2 -od ~/Desktop/ -  
of sqematest_R>1
```

This identifies all the samples with only one OTU and deletes them:

Low-R samples removed:

```
100198,100204,100213,100219,100257,100260,100262,100266,100271,1  
00272,100279,100282,100285,100287,100290,100291,100293,100297,10  
0306,
```

and saves a file on the Desktop called sqematest_R>1.csv with the results in it.

filter_keep takes the name of OTUs present in one specified column, then keeps all OTUs/rows that have these. This is useful for filtering OTU tables so that only one taxonomic group is retained. For example, if you have metabarcoding data from a primer set that amplifies from all Eukarya, but want to analyse only results for two phyla, specify the phyla of interest with **-fi**, and the column where these phyla are listed with **-fc**.

-fi --filter_items

Type = str, default = "

Names as a spaceseparated list to be used as a filter for keeping rows in an OTU table.

-fc --filter_column

Type= str default = ''

Title of column containing filter for losing rows in an OTU table.

Example:

An OTU table with full taxonomic information and some sequencing information is included with sqema, SharkBay16S_HP_1.csv. To focus on analysis of just one taxonomic group, for example Actinopteri (bony fish), this command will remove all other OTUs:

```
python sqema.py filter_keep -in SharkBay16S_HP_1.csv -fc class -  
fi Actinopteri
```

Result:

This removes all OTUs that don't belong to class Actinopteri as shown in the "class" column of the datasheet from the lab. The new .csv file is written to the default seqma_output/ directory.

filter_lose does the opposite of **filter_keep**. This can be used to remove taxonomic groups that are not to be analysed from an OTU table. An example application might be to reduce OTUs that are off-target amplifications commonly found with eDNA metabarcoding.

-fi --filter_items

Type = str, default = " (empty string)

Names as a space-separated list to be used as a filter for removing rows in an OTU table.

-fc --filter_column

Type= str default = ''

Title of column containing filter for losing rows in an OTU table.

Example:

The file SharkBay16S_HP_1.csv was generated with a metabarcoding primer set targeting fish, but it also amplifies some bacterial and archaeal DNA. To remove bacterial and archaeal OTUs, use the - **filter_lose** command:

```
python sqema.py filter_lose -in SharkBay16S_HP_1.csv -fc domain  
-fi Bacteria Archaea
```

Result:

A new .csv file is written to the seqma_output default folder that only has OTUs from Eukaryota retained.

merge_columns combines names across columns and places them in one column. This is used for editing OTU tables with multi-column taxonomies when it is useful to retain several levels of taxonomic information.

-mt --merge_titles

Type=str,default

Name for column/samples to be combined when merging.

-nn --new_name

Type=str,default

Name for new column/sample when combined.

Example:

This can be used for combining OTU names into a single column when they are presented in separate columns for providing taxonomic and technical information. As an example, the SharkBay16S_HP_1.csv file contains genus, species and OTU information that could be combined to make a one-column name. This command does this:

```
python sqema.py merge_columns -in SharkBay16S_HP_1.csv -mt genus  
species zotu -nn GenSpZOTU
```

Result:

A new .csv file is written where the first column contains genus:species:OTU names concatenated. The other columns could be removed using **rm_columns** to make the OTU table useable for sqema quantitative analyses.

2.4 OTU table row/OTU modification

rm_rows removes OTUs as a whole row from an OTU table.

-b --basis

Type = choices= names, POO, low_quant, default = 'names'.

Defines the basis for removing samples, either by names, proportion of occurrence in the OTU table, or low relative quantification based on an SAD.

-rn --rm_names

Type = str, default = "

If "names" is specified for **-basis**, the names of OTUs (rows) to remove from an OTU table.

-pt --POO_threshold

Type = float, default = 0.005

Minimum POO for OTUs to be retained in the OTU table.

-qt -quant_threshold

Type = float, default = 0.001

Minimum estimated biomass proportion for an OTU to be retained.

Example:

To remove OTUs that are present at less than 1% estimated proportional quantity from the default datafile, with default parameters for an SAD, this is the command:

```
python sqema.py rm_rows -b low_quant -qt 0.01
```

Result:

This saves a reduced .csv OTU table file in the default directory and reports which OTUs were removed:

Based on SAD model: logser

Metric: RRA

Transform: untransf

With a threshold for inclusion of > 0.01 estimated total proportion in the OTU table

OTUs removed: Aiptasiogeton eruptaurantia,Obeliida sp.3,Obelia sp.1,Edwardsia longicornis,Bougainvillia sp. ,Coryne eximia,Stauridiosarsia marii,Leuckartiara cf. octonema PS-2018,Turritopsis sp.1,Turritopsis dohrnii,Obeliida

sp.2, Eudendrium carneum, Zanclea migottoi, Clytia sp.2, Zancleopsis dichotoma, Anthopleura elegantissima,

merge_otus merges two or more rows of an OTU table. Reads are summed for each sample. The names of the OTUs are combined and renamed with the **-new_OTU_name** command, or if combined automatically if a new name is not specified. This command is useful for merging multiple OTUs from one species into a single OTU representing all DNA variants in the species.

-mt --merge_titles

Type = str, default = '' (empty string)

Name for rows/OTUs to be combined when merging.

-nn --new_OTU_name

Type = str, default = '' (empty string)

Name for a new row/OTUs when merging columns.

-fc --filter_column

Type = str default = ''

Title of column containing filter for merging rows in an OTU table. If **-fc** is not specified, sqema will default to the first column in the OTU table.

Example:

```
python sqema.py merge_otus -mt Clytia_sp.1 Clytia_sp.2 -nn  
Clytia_sp
```

Result:

This combines two OTUs in the default sqematest_t.csv file. The output to the terminal is:

OTUs Clytia sp.1, Clytia sp.2, combined into new OTU Clytia_sp with reads summed for each sample

The new .csv file has the merged OTU at the end of the table, and Clytia sp. 1 and Clytia sp. 2 have been removed. Their read counts were summed for each sample and entered in the same sample columns in Clytia_all. Note that Clytia gracilis is retained as a separate OTU as it was not specified that it should be combined.

ScientificName	100162	100163	100164	100165
Campanularia hincksii	1496	5075	53	1502
Podocorynoides minima	1024	565	0	94

Turritopsis rubra	0	0	0	0
Bougainvillia muscus	901	849	189	5
Aequorea macrodactyla	0	0	0	0
Orthopyxis crenata	136	66	63	0
Obelia dichotoma	59	184	177	0
Clytia gracilis	0	0	0	0
Actiniaria sp.	0	309	0	0
Liriope tetraphylla	0	0	0	0
Aiptasiogeton eruptaurantia	0	0	0	0
Coryne eximia	142	0	0	5
Bougainvillia sp.	0	0	0	456
Obelia sp.1	47	231	33	0
Obeliida sp.2	0	0	0	0
Obeliida sp.3	0	0	0	0
Turritopsis dohrnii	0	0	0	0
Turritopsis sp.1	85	0	0	0
Edwardsia longicornis	0	0	0	0
Eudendrium carneum	0	0	0	0
Stauridiosarsia marii	0	0	0	0
Leuckartiara cf. octonema PS-2018	0	0	0	0
Zancklea migottoi	0	0	0	0
Anthopleura elegantissima	0	0	0	5
Zanckleopsis dichotoma	0	0	0	0
Clytia_sp	0	0	0	0

merge_otus_auto merges rows of an OTU table that share an entry in column **-filter_column**. This is useful for grouping multiple ZOTUs/ASVs to species level, for example. In this case, given an OTU table with a column "species", enter that title via **-fc**. The function will search for all shared species names in the column, and merge the rows that have them, summing reads in the numerical columns.

-fc --filter_column

Type= str default = ''

Title of column containing filter for merging rows in an OTU table. If **-fc** is not specified, sqema will default to the first column in the OTU table.

Example:

```
Python sqema.py merge_otus_auto -in SharkBay16S_EG_1.csv -fc  
species -of SharkBay16S_EG_1_species_merged
```

2.5 Simulation

sim_samples creates a simulated eDNA metabarcoding dataset. This is useful for experimental planning, or for exploring the value of using the SAD fitting that SQEMA implements for quantification in eDNA metabarcoding data. This function takes a user-defined number of species from a distribution defined by the user. Sampling is repeated to simulate multiple random samples from the same population. Patchiness of eDNA detection is simulated by a user-defined parameter. Results are written to a .csv formatted file as an OTU table. A separate file is written with dispersal metrics recorded for the simulated dataset. These can be used to compare dispersion among distributions and might be helpful as a basis for making a simulated OTU table with dispersion of values from the SAD similar to that seen in real data (see section 2.4 – dispersion metrics).

Parameters:

- sp --shape_par**
type = float, default = 0.9
Shape metric for defining an SAD model of expected counts.
- pch --patchiness**
type = float, default = 0.1
Proportion of OTUs in a simulated sample lost due to eDNA patchiness.
- simR --sim_richness**
type = int, default = 25
Number of species in a simulated community.
- simC --sim_counts**
type = int, default = 10000
Counts (sequencing reads) per sample in a simulated community.
- simN --sim_samples**
type = int, default = 25
Number of samples for an OTU table derived from a simulated community.

Example:

```
python sqema.py sim_samples -simR 50 -simN 100 -simC 30000 -sp  
0.88 -pch 0.2
```

Result:

One .csv OTU table with 50 species, 100 samples, 3000 reads per sample, a 0.2 proportion loss of items per sample (the zeroes are largely from this. This is a portion of the table:

	Sample_1	Sample_2	Sample_3	Sample_4	Sample_5
Species_1	12413	12434	0	12218	12385
Species_2	5585	5435	17963	0	0
Species_3	3217	3215	0	8707	8799
Species_4	2175	2084	5331	2130	2067
Species_5	1523	1511	1476	1547	1512
Species_6	1059	1141	0	1156	1121
Species_7	845	849	1965	0	0
Species_8	553	0	651	1470	1443
Species_9	478	1141	469	0	0
Species_10	384	368	0	923	502

A .csv file is also produced showing measures of dispersion for the OTU table, a portion of which looks like this:

Measures of dispersion for simulated OTU table

Species richness	50
Samples	100
Counts per sample	30000

Dataset means

mean MAE	220.04
mean MSE	1638972.94
mean RMSD	871.65227

Sample dispersions

	1	2	3	4	5
MAE	523.183673	532.142857	70.4285714	65.7346939	54.3877551
MSE	6323072.12	6311362.59	36395.3673	40064.5102	30861.6122
RMSD	2514.57196	2512.24254	190.775699	200.161211	175.674734

2.6 Community comparisons

venn draws a Venn diagram for overlap and independence of OTUs in two communities. The input is a pair of .csv files with the title of a column of names (the same in both files) indicated by the **-cn** parameter. A scaling factor for the Venn diagrams can be used to adjust the output where more extreme values cause the text to not fit the overlaps well. The default value of 2 is generally effective, 1 works for small R values, and > 2 if there is extensive overlap between communities. The value is decimal and changes the size of the overall graph, while the text stays the same. The graph indicates counts of OTUs in bold, and proportions in normal type under that.

The file output is .svg only and can be determined with the **-od** and **-of** parameters, or the default file will go to the **sqema_output** folder.

Parameters:

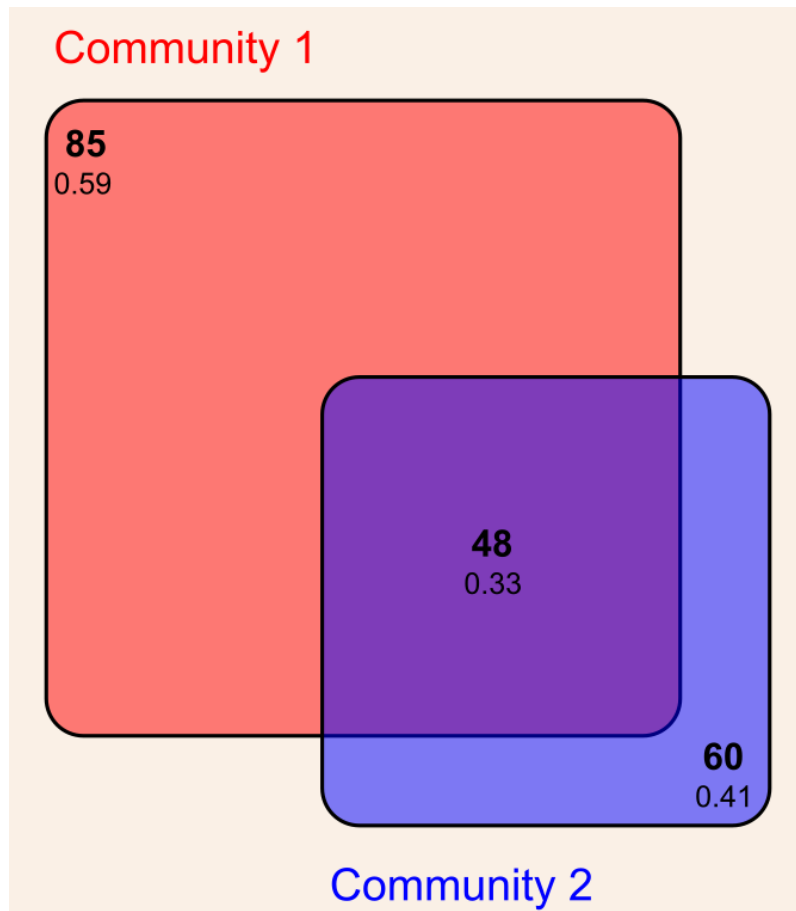
```
-cf1 --comparison_file_1  
    type = str, default = none  
-cf2 --comparison_file_2  
    type = str, default = none  
-cn --comparison_names  
    type = string, default = none  
-gs --graph_scale  
    type=float, default = 2
```

Example:

```
python3 sqema.py venn -cf1 python sqema.py venn -cf1  
~/Sqema/SB_data/SharkBay16S_EG_11.csv -cf2  
~/Sqema/SB_data/SharkBay16S_HP_8.csv -cn GenSpZOTU
```

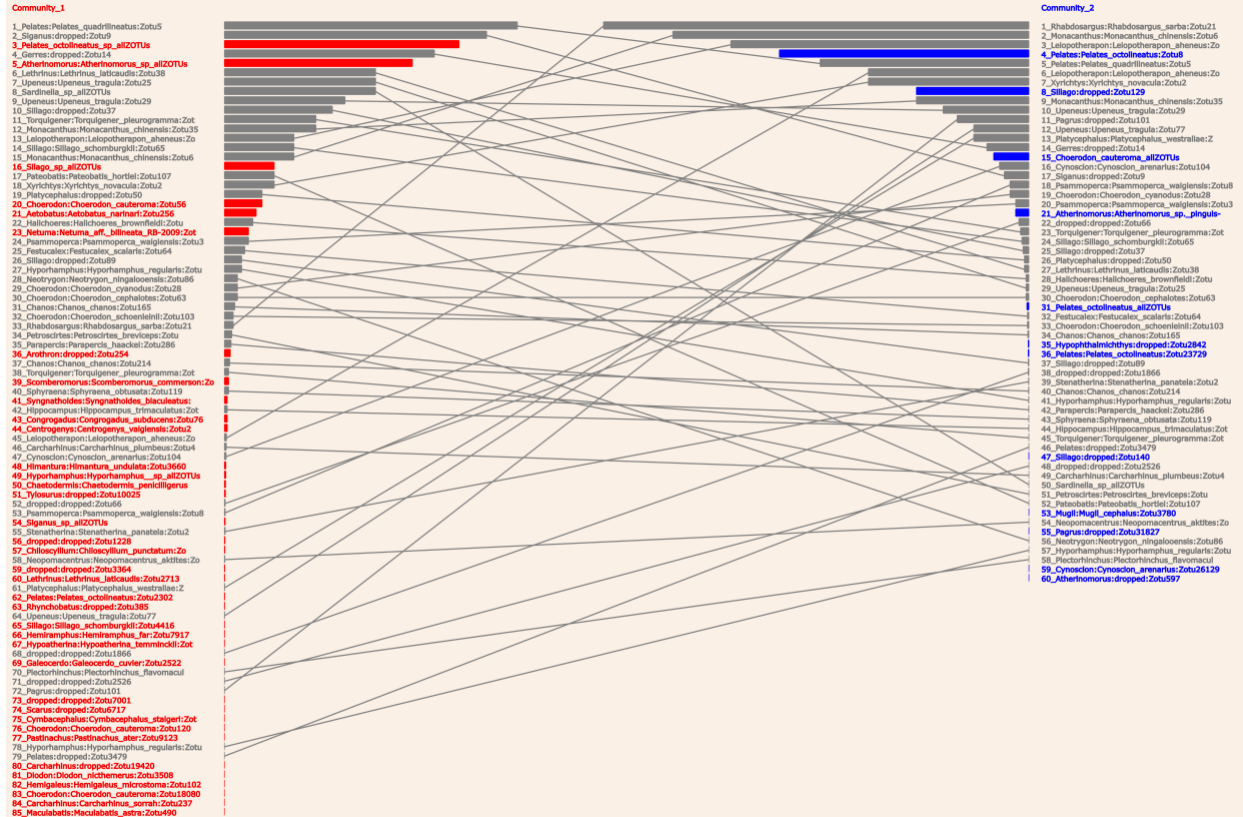
Result:

This plot is produced:



... which is saved in the `sqema_output/` directory. These plots use squares with sizes proportional to the species richness in each community, and an overlap that is proportional to that as well.

`arch_plot` draws a diagram that compares the abundance of species in two communities and links abundance between species that occur in both, like this:



where items unique to each community have a red or blue indication and the shared items are grey and a line links their entry for each community. The inputs for this are two .csv files with columns of OTU names and their estimated abundance proportions in them, as is produced by the output of [quantify](#). The names of the headers for the columns should be the same in each file, and have to be specified in the input for this function.

Parameters:

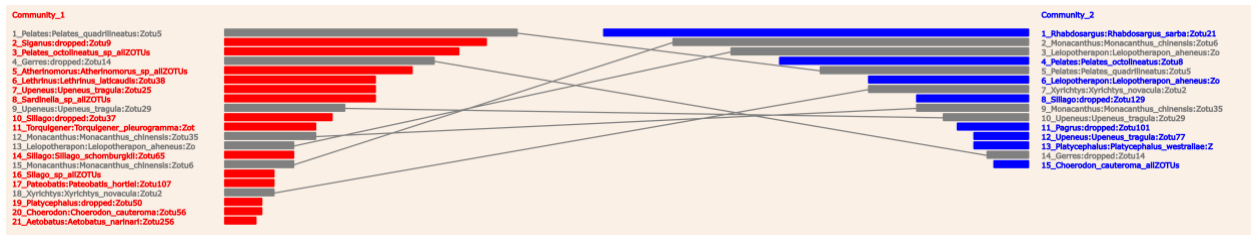
- cf1 --comparison_file_1**
type = str, default = none
- cf2 --comparison_file_2**
type = str, default = none
- cn --comparison_names**
type = float, default = 0.0
- cp --comparison_proportions**
type = float, default = 0.0
- qt --quant_threshold**
type = float, default = 0.0

Example:

```
python sqema.py arch_plot -cf1
~/Dropbox/Wild_Frontiers/sqema/SharkBay16S_EG_quantalluvial.csv
-cf2
~/Dropbox/Wild_Frontiers/sqema/SharkBay16S_HP_quantalluvial.csv
-cn Names -cp Proportions -qt 0.01
```

Result:

This produces a plot of fish species in two adjacent communities in the Shark Bay area of Western Australia, with only species present at greater than 1% of the total included:



2.7 Dispersion metrics

Measures of dispersal of data from an underlying distribution are useful for measuring the fit of real or simulated data to a model.

dispersions is the base command. This takes an OTU table as input, as well as model parameters for an SAD. The degree of dispersion on the data around the expected values for the SAD is measured and reported as:

MAE mean absolute error (https://en.wikipedia.org/wiki/Mean_absolute_error)

MSE mean squared error (https://en.wikipedia.org/wiki/Mean_squared_error)

RMSD root mean squared deviation

(https://en.wikipedia.org/wiki/Root_mean_square_deviation)

-mo --model

Type=choices, options = logser, zipf, genpareto, lognorm , geom , linear, nbinom, default = logser

The SAD model used to determine expected counts

-sp --shape_par

type=float, default=0.

Definesthe shape of an SAD to determine expected counts.

Example:

```
python sqema.py dispersions -in sqematest_1.csv -mo zipf -sp 3
```

Result:

A .csv file is written giving measures of dispersion for the OTU table based on the specified SAD, a portion of which looks like this:

Model:	zipf	Shape parameters:	3	
Cumulative frequency distribution:				
0.8319073725807077	0.93589579	0.96670718	0.97970573	0.98636099

OTU table mean dispersions

mean MAE	48.9399038
mean MSE	81987.5871
mean RMSD	147.783538

Sample dispersions

Sample	100162	100163	100164	100165
MAE	146.807692	114.038462	34.6153846	35.3461538
MSE	161452.038	68938.9615	8194.84615	10543.9615
RMSD	401.810949	262.5623	90.5253896	102.683794

2.8 OTU table biodiversity metrics

bdiv_metrics calculates a range of metrics and standard ecological indices from the OTU table (https://en.wikipedia.org/wiki/Diversity_index). The metrics are written to a CSV file with the following lines:

Sample names - the names of the samples in the first row of the columns of the .csv formatted OTU table input file.

N reads - the total count of sequence reads in each sample of the OTU table input file.

Species richness (R or S) - the number of unique OTUs in the sample.

Simpson index (λ) - the probability that any two sequence reads are from the same species (Simpson 1949).

Gini-Simpson index ($1-\lambda$) - the probability of encounter of two species present in a samples (Hurlbert 1971).

Shannon index (H') - index representing the chance that an OTU in a sample will be novel rather than previously encountered (Shannon 1948).

Berger-Parker index - the proportion of sample reads for the most abundant OTU.

Hill D0 - value of D from the General Equation of Diversity with q set to 0 - identical to Species Richness (Alberdi and Gilbert 2019).

Hill D1 - value of D from the General Equation of Diversity with q set to 1 - equal to 1/Simpson index.

Hill D2 - value of D from the General Equation of Diversity with q set to 2 - equal to exp (Shannon Index).

Example:

```
python sqema.py bdiv_metrics -in sqematest_1.csv
```

Result:

A .csv file is written that looks like this:

Sample	100162	100163	100164	100165	100166
N reads	3890	3890	7279	7279	515
Richness (R)	8	7	5	6	2
Simpson index (lambda)	0.27424931	0.50926291	0.28246583	0.57878464	0.81440544
Gini-Simpson index	0.72575069	0.49073709	0.71753417	0.42121536	0.18559456
Shannon-Weaver diversity (H')	1.49614544	1.07969831	1.40203797	0.74970922	0.33273184

Berger-Parker index	0.38457584	0.69721116	0.36699029	0.72665699	0.89648799
Hill D0	8	7	5	6	2
Hill D1	4.46444736	2.94379131	4.06347279	2.11638452	1.39477322
Hill D2	3.64631727	1.96362229	3.54025121	1.72775836	1.22788963

3.0 Example workflow

Shark Bay is a large, shallow feature on the coast of Western Australia. It features a strong salinity gradient from hypersaline waters in Hamelin Pool at the furthest point from the sea, grading down to globally average marine salinity at the ocean mouth of the bay.



eDNA metabarcoding data from two regions of Shark Bay, Hamelin Pool and the Eastern Gulf, are included as sample data. These are included with sqema in files:

```
SharkBay16S_HP_1.csv  
SharkBay16S_EG_1.csv
```

Here is an example workflow for the question:

What differences are there between the most abundant fish species found in Hamelin Pool and in the Eastern Gulf of Shark Bay?

We will use the eDNA metabarcoding data, which was collected by sampling the sea-bottom biofilms with paint rollers on a pole (Jarman et al. 2024). This sampling method produces a good overall assessment of fish biodiversity in these conditions, with many benthic species detected that water column sampling does not (Richards et al., submitted).

To start, we will filter the file to include only data from fish. The classes Actinopteri and Chondrichthyes contain all the fish in the data, so we will use the `filter_keep` command to only retain OTUs from that class. The new file will be written to a folder "SB_data" on the Desktop:

```
python sqema.py filter_keep -in SharkBay16S_HP_1.csv -fc class -  
fi Actinopteri Chondrichthyes -od ~/Desktop/SB_data/ -of  
SharkBay16S_HP_2
```

Each OTU needs a unique name in one column, so combining the genus, species, and name columns is a good option with:

```
python sqema.py merge_columns -in  
~/Desktop/SB_data/SharkBay16S_HP_2.csv -mt genus species OTU -nn  
GenSpZOTU -od ~/Desktop/SB_data/ -of SharkBay16S_HP_3
```

Which loads the file from the folder made in step 1. The next step is to remove columns that we won't need for the analyses:

```
python sqema.py rm_columns -rn domain phylum class order family  
numberOfUnq_BlastHits -in ~/Desktop/SB_data/SharkBay16S_HP_3.csv  
-od ~/Desktop/SB_data/ -of SharkBay16S_HP_4
```

At this point, the file SharkBay16S_HP_4.csv is formatted so that it could be used for sqema quantitative analyses, as an OTU table with the first column containing OTU names, and the first row containing sample names, and it looks like this:

GenSpZOTU	E_442_001	E_442_002	E_442_003	E_442_004
Xyrichtys:Xyrichtys_novacula:Zotu2	0	0	0	0
Pelates:Pelates_quadrilineatus:Zotu5	1	2	0	2
Monacanthus:Monacanthus_chinensis:Zotu6	0	5	2	1
Pelates:Pelates_octolineatus:Zotu8	1	9	1	0
Siganus:dropped:Zotu9	0	0	1	6
Leiopotherapon:Leiopotherapon_aheneus:Zotu11	0	5	0	11
Gerres:dropped:Zotu14	2	5	0	6
Rhabdosargus:Rhabdosargus_sarba:Zotu21	6	11	9	3

We should remove samples with low total read numbers for the fish groups of interest so that we don't bias the results too much, which we can do with the command:

```
python sqema.py rm_columns -b low_reads -minReads 200 -in
~/Desktop/SB_data/SharkBay16S_HP_4.csv -od ~/Desktop/SB_data/ -
of SharkBay16S_HP_5
```

Which removes any sample with less than 200 reads from the OTU table. At this point, the data is ZOTUs and there are three putative species represented by multiple ZOTUs in the data:

Choerodon:Choerodon_cauteroma:Zotu56
Choerodon:Choerodon_cephalotes:Zotu63
Choerodon:Choerodon_schoenleinii:Zotu103
Choerodon:Choerodon_cauteroma:Zotu120

Sardinella:dropped:Zotu74
Sardinella:dropped:Zotu76

Pelates:Pelates_octolineatus:Zotu236
Pelates:Pelates_octolineatus:Zotu2302
Pelates:Pelates_octolineatus:Zotu31462
Pelates:Pelates_octolineatus:Zotu33495

For a stringent analysis, these should be combined so that we get as close to species-level as possible.

```
python sqema.py merge_otus -in
~/Desktop/SB_data/SharkBay16S_HP_5.csv -od ~/Desktop/SB_data/ -
of SharkBay16S_HP_6 -mt Choerodon:Choerodon_cauteroma:Zotu56
Choerodon:Choerodon_cauteroma:Zotu63
Choerodon:Choerodon_cauteroma:Zotu103
Choerodon:Choerodon_cauteroma:Zotu120
Choerodon:Choerodon_cauteroma:Zotu74
Choerodon:Choerodon_cauteroma:Zotu76 -nn
Choerodon_cauteroma_allZOTUs
```

```
python sqema.py merge_otus -in
~/Desktop/SB_data/SharkBay16S_HP_6.csv -od ~/Desktop/SB_data/ -
of SharkBay16S_HP_7 -mt Sardinella:dropped:Zotu74
Sardinella:dropped:Zotu76 -nn Sardinella_sp_allZOTUs
```

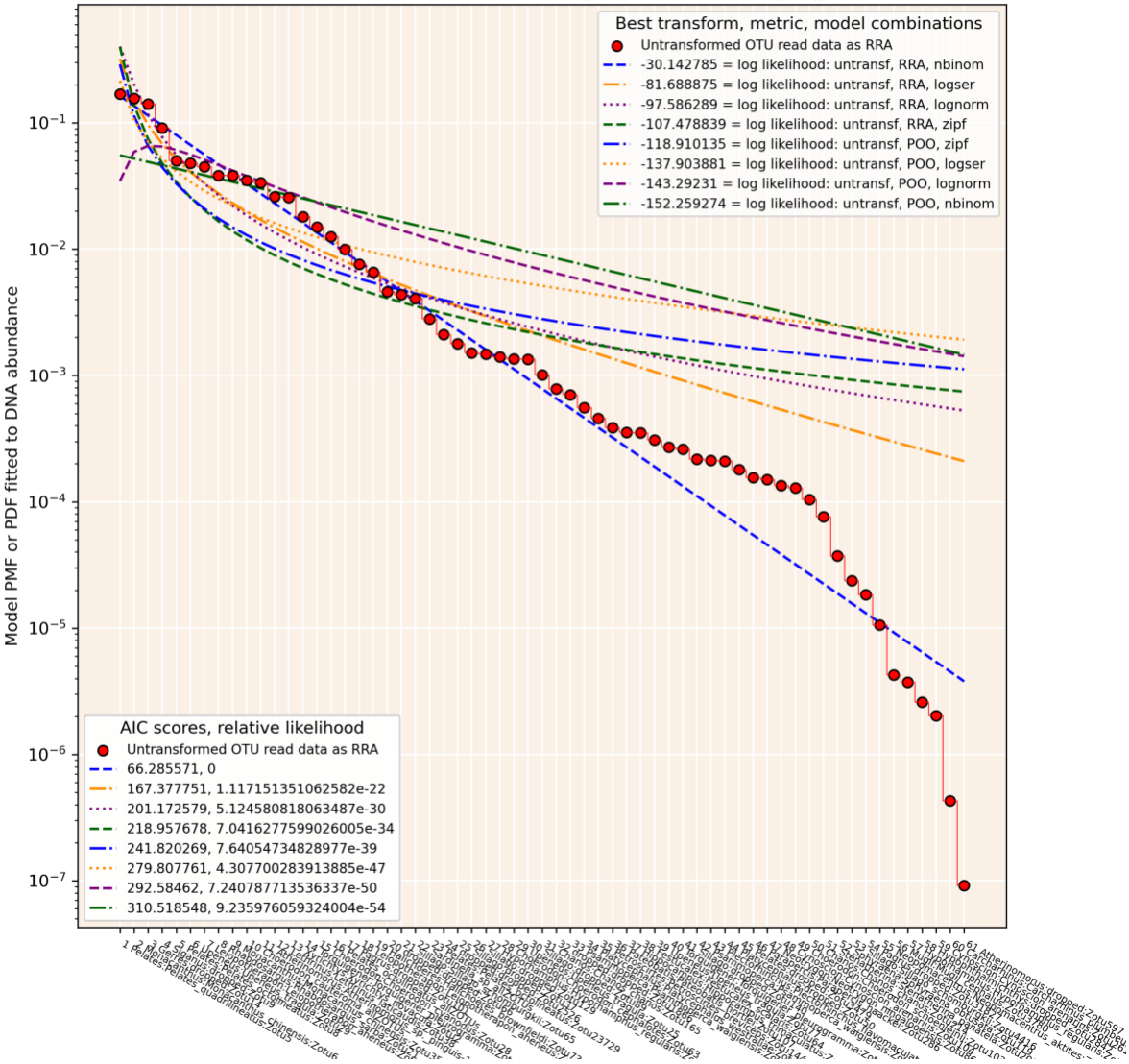
```
python sqema.py merge_otus -in
~/Desktop/SB_data/SharkBay16S_HP_7.csv -od ~/Desktop/SB_data/ -
of SharkBay16S_HP_8 -mt Pelates:Pelates_octolineatus:Zotu236
Pelates:Pelates_octolineatus:Zotu2302
Pelates:Pelates_octolineatus:Zotu31462
Pelates:Pelates_octolineatus:Zotu33495 -nn
Pelates_octolineatus_allZOTUs
```

The file SharkBay16S_HP_8.csv now has one column of taxonomic information, which corresponds to species as closely as we can make it. To find a good SAD model for the dataset, this command is used:

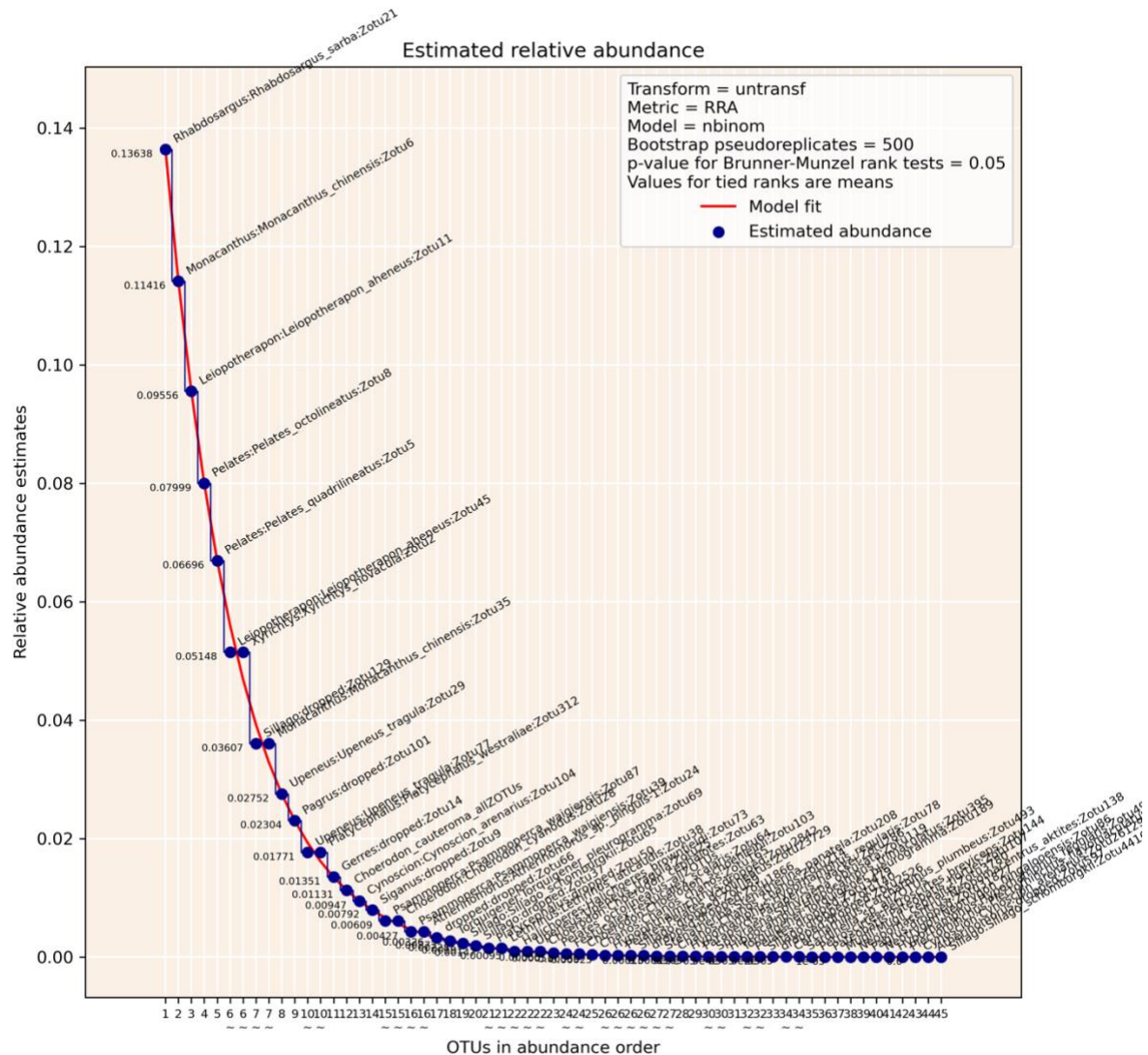
```
python sqema.py fit_SAD -in
~/Desktop/SB_data/SharkBay16S_HP_8.csv -od ~/Desktop/SB_data/
```

which shows that the negative binomial model with relative read abundance is the best fitting option tested:

Fitting for /Users/273915i/Desktop/SB_data/SharkBay16S_HP_8.csv



```
python sqema.py quantify -mo nbinom -py linear -in
~/Desktop/SB_data/SharkBay16S_HP_8.csv -od ~/Desktop/SB_data/
```



Following the same steps for the Eastern Gulf samples up to EG_5, we get an OTU table with these duplicates:

Sardinella:dropped:Zotu74
Sardinella:dropped:Zotu76
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu12852
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu13388
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu9363
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu12852
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu13388
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu16069
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu16069
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu29684
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu29684
Hyporhamphus:Hyporhamphus_quoyi:Zotu8696
Hyporhamphus:Hyporhamphus_quoyi:Zotu8798
Choerodon:Choerodon_cyanodus:Zotu28
Choerodon:Choerodon_cyanodus:Zotu28
Choerodon:Choerodon_schoenleinii:Zotu103
Choerodon:Choerodon_cauteroma:Zotu120
Choerodon:Choerodon_cauteroma:Zotu56
Choerodon:Choerodon_cephalotes:Zotu63
Pelates:Pelates_octolineatus:Zotu23729
Pelates:Pelates_octolineatus:Zotu31462
Pelates:Pelates_octolineatus:Zotu33495
Pelates:Pelates_octolineatus:Zotu236
Pelates:Pelates_octolineatus:Zotu8
Pelates:Pelates_octolineatus:Zotu236
Siganus:dropped:Zotu5100
Siganus:dropped:Zotu18538
Siganus:dropped:Zotu24429
Siganus:dropped:Zotu26039
Sillago:dropped:Zotu129
Sillago:dropped:Zotu140

To combine these Multi-ZOTU groups into single putative OTUs, we can use the series of commands below:

```
Sqema merge_otus -in ~/Desktop/SB_data/SharkBay16S_EG_5.csv -od  
~/Desktop/SB_data/ -of SharkBay16S_EG_6 -mt
```

```
Sardinella:dropped:Zotu74 Sardinella:dropped:Zotu76 -nn
Sardinella_sp_allZOTUs
```

```
python sqema.py merge_otus -in
~/Desktop/SB_data/SharkBay16S_EG_6.csv -od ~/Desktop/SB_data/ -
of SharkBay16S_EG_7 -mt Atherinomorus:Atherinomorus_sp._punguis-
1:Zotu12852 Atherinomorus:Atherinomorus_sp._punguis-1:Zotu13388
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu9363
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu12852
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu13388
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu16069
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu16069
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu29684
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu21345
Atherinomorus:Atherinomorus_sp._punguis-1:Zotu24 -nn
Atherinomorus:Atherinomorus_sp_allZOTUs
```

```
python sqema.py merge_otus -in
~/Desktop/SB_data/SharkBay16S_EG_7.csv -od ~/Desktop/SB_data/ -
of SharkBay16S_EG_8 -mt Hyporhamphus:Hyporhamphus_quoyi:Zotu8696
Hyporhamphus:Hyporhamphus_quoyi:Zotu8798 -nn
Hyporhamphus:Hyporhamphus__sp_allZOTUs
```

```
python sqema.py merge_otus -in
~/Desktop/SB_data/SharkBay16S_EG_8.csv -od ~/Desktop/SB_data/ -
of SharkBay16S_EG_9 -mt Pelates:Pelates_octolineatus:Zotu23729
Pelates:Pelates_octolineatus:Zotu31462
Pelates:Pelates_octolineatus:Zotu33495
Pelates:Pelates_octolineatus:Zotu236
Pelates:Pelates_octolineatus:Zotu8
Pelates:Pelates_octolineatus:Zotu236 -nn
Pelates_octolineatus_sp_allZOTUs
```

```
python sqema.py merge_otus -in
~/Desktop/SB_data/SharkBay16S_EG_9.csv -od ~/Desktop/SB_data/ -
of SharkBay16S_EG_10 -mt Siganus:dropped:Zotu5100
```

```
Siganus:dropped:Zotu18538 Siganus:dropped:Zotu24429
Siganus:dropped:Zotu26039 -nn Siganus_sp_allZOTUs
```

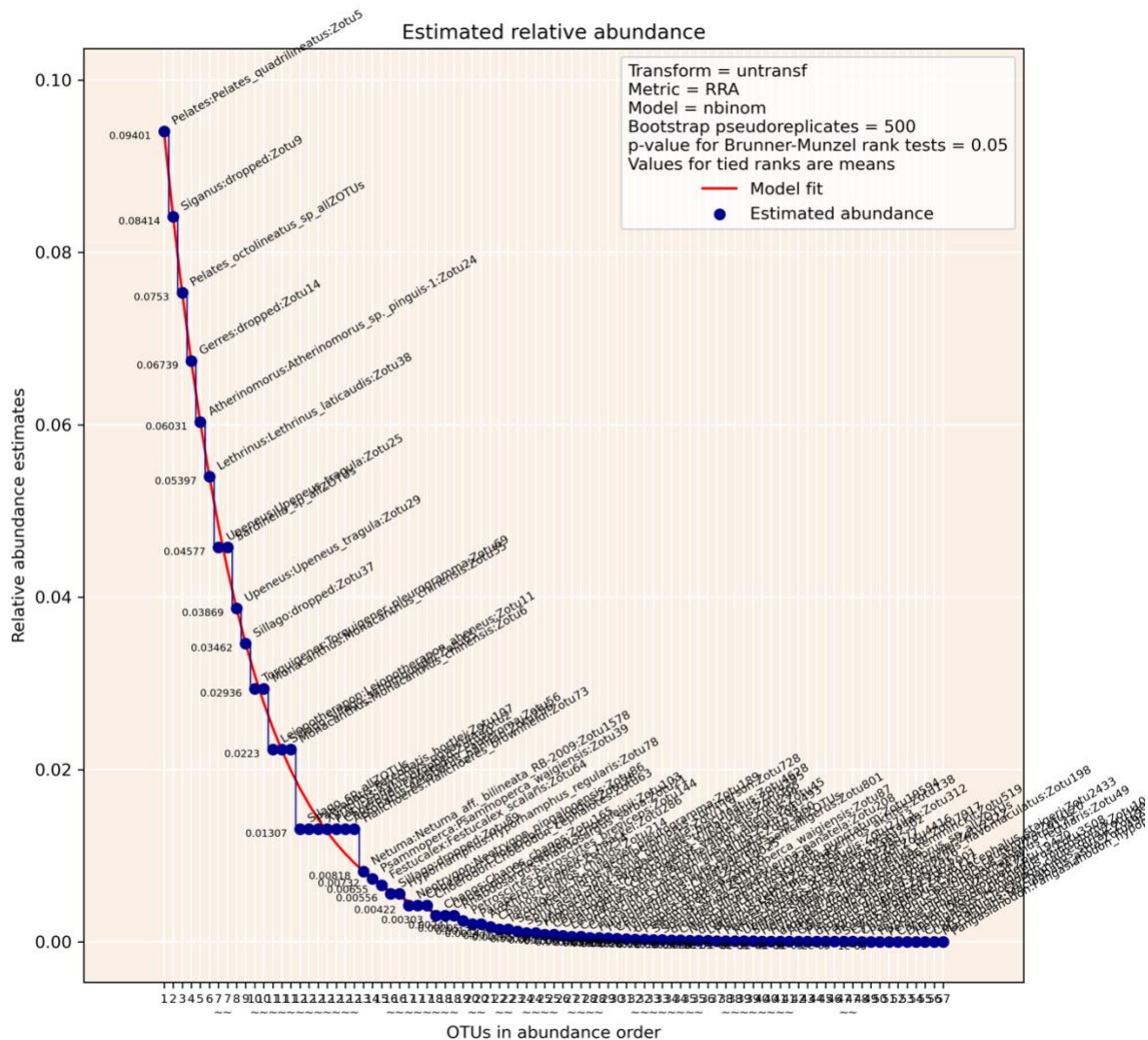
```
python sqema.py merge_otus -in
~/Desktop/SB_data/SharkBay16S_EG_10.csv -od ~/Desktop/SB_data/ -
of SharkBay16S_EG_11 -mt Sillago:dropped:Zotu129
Sillago:dropped:Zotu140 -nn Silago_sp_allZOTUs
```

File SharkBay16S_EG_11.csv is now equivalent to SharkBay16S_HP_8.csv and the same quantitative analyses can be done with it, e.g:

```
python sqema.py quantify -mo nbinom -py linear -in
~/Desktop/SB_data/SharkBay16S_EG_11.csv -od ~/Desktop/SB_data/
```

If we compare the .csv file quantitative, there are some overlaps in fish species, but the ten most abundant fish are completely different. These comprise approximately 60% of the estimated biomass in each community:

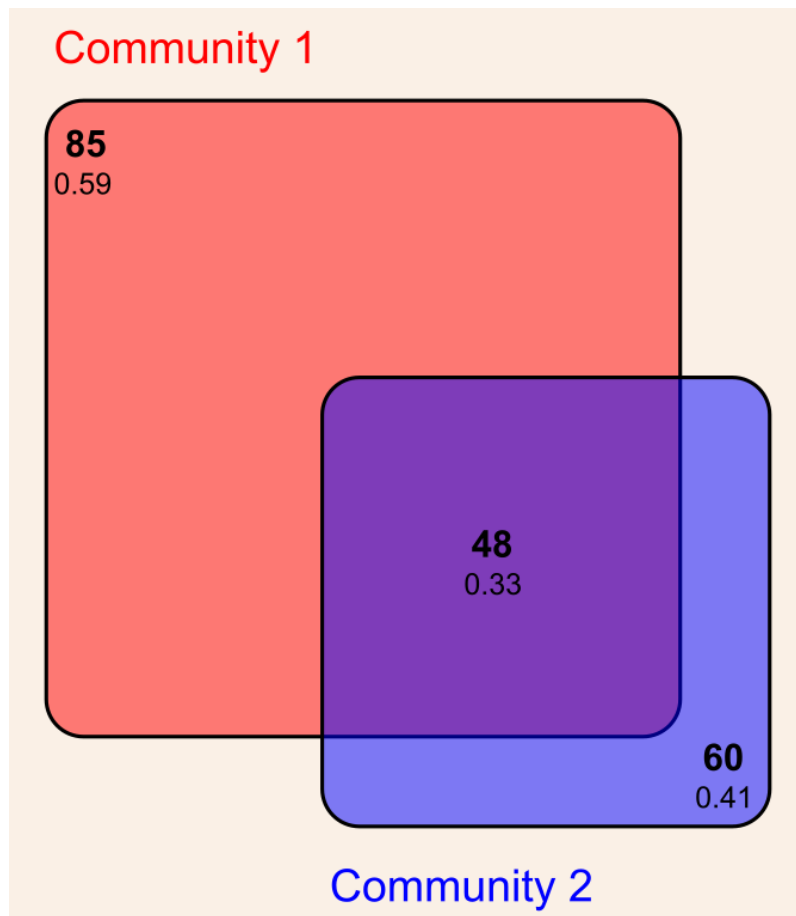
EG_OTUs ranked by abundance	EG Relative abundance with means for tied ranks	HP_OTUs ranked by abundance	HP_Relative abundance with means for tied ranks
Pelates: Pelates_quadrilineatus: Zotu5	0.094014847	Rhabdosargus: Rhabdosargus_sarba: Zotu21	0.13638101
Siganus: dropped: Zotu9	0.084138546	Monacanthus: Monacanthus_chinensis: Zotu6	0.11416102
Pelates: octolineatus_sp_allZOTUs	0.075299754	Leiopotherapon: Leiopotherapon_aheneus: Zotu11	0.09556124
Gerres: dropped: Zotu14	0.067389482	Pelates: Pelates_octolineatus: Zotu8	0.07999184
Atherinomorus: Atherinomorus_sp._punguis-1: Zotu24	0.060310187	Pelates: Pelates_quadrilineatus: Zotu5	0.06695911
Lethrinus: Lethrinus_laticaudis: Zotu38	0.053974575	Leiopotherapon: Leiopotherapon_aheneus: Zotu45	0.05148376
Upeneus: Upeneus_tragula: Zotu25	0.045767316	Xyrichtys: Xyrichtys_novacula: Zotu2	0.05148376
Sardinella_sp_allZOTUs	0.045767316	Sillago: dropped: Zotu129	0.03607432
Upeneus: Upeneus_tragula: Zotu29	0.038688768	Monacanthus: Monacanthus_chinensis: Zotu35	0.03607432
Sillago: dropped: Zotu37	0.034624496	Upeneus: Upeneus_tragula: Zotu29	0.02751879



The two fish communities are unsurprisingly different in composition, being in completely different salinity zones despite being close to each other. However, the extent of the difference would be masked if only incidence data was used, which is common in eDNA studies. There Venn diagram below shows that almost half of the species are shared, whereas the biomass composition is very different.

The diagram is generated by:

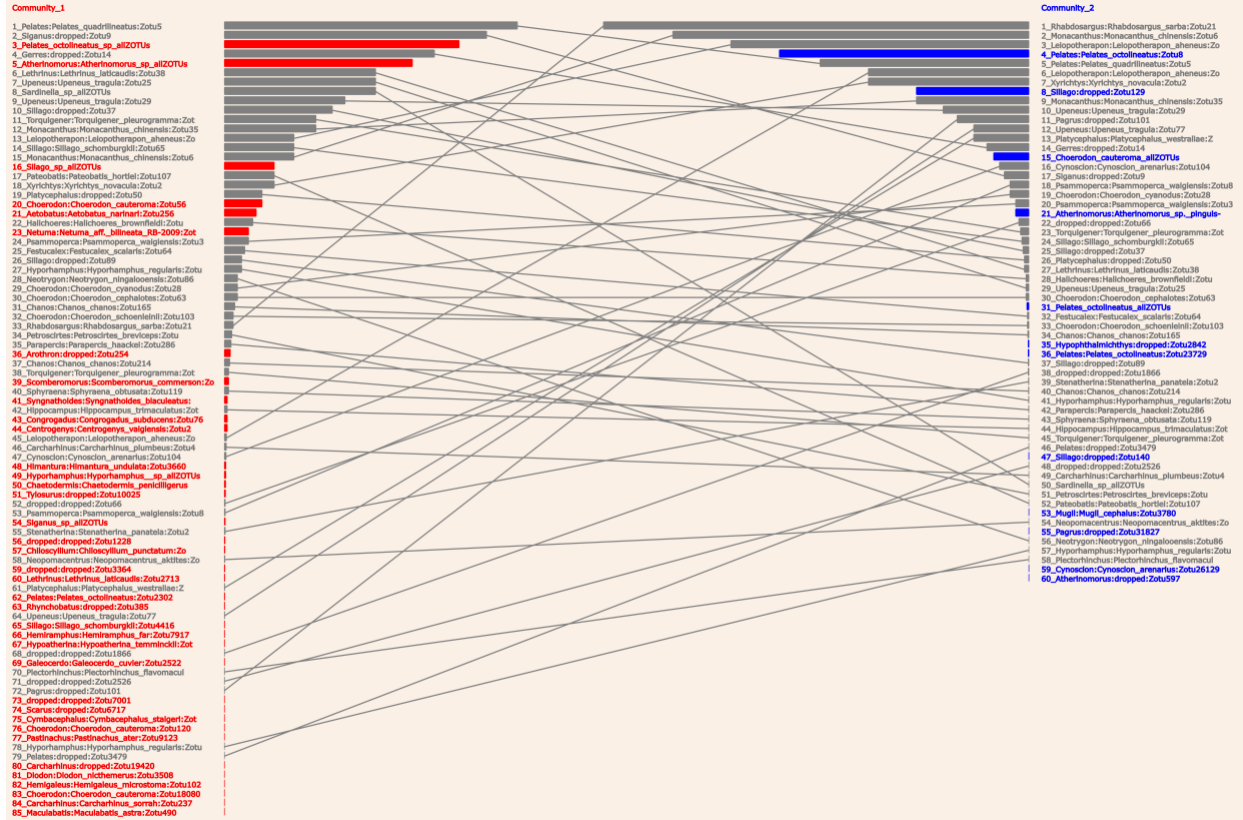
```
python sqema.py venn -cf1 ~/sqema/SB_data/SharkBay16S_EG_11.csv
-cf2 ~/sqema/SB_data/SharkBay16S_HP_8.csv -cn GenSpZOTU
```

If we consider abundance, however, a more realistic picture of two separate fish communities can be seen. This command:

```
python sqema.py arch_plot -cf1  
~/Desktop/SB_data/SharkBay16S_EG_11_quantify_scores.csv -cf2  
~/Desktop/SB_data/SharkBay16S_HP_8_quantify_scores.csv -cn 'OTUs  
ranked by abundance' -cp 'Estimated relative abundance'
```

Produces a plot of the abundance of OTUs in both communities. Shared OTUs are shown in grey, with links between the entries in each community given as a diagonal line. OTUs unique to each community are indicated in red or blue, as shown here:



What this shows is that the communities are very different when abundance is considered. The high biomass species are quite different in each region, but in many cases the high biomass species from one community shows up as a low biomass species in the adjacent community, so the Venn diagram view of extensive community overlap based on incidence is misleading.

4.0 Command list

Base functions are the only positional argument follows the "sqema" command and tells it what to do. These do not require a "-" or "--". The options are:

`fit_SAD quantify bdiv_metrics dispersions sim_samples
rm_columns rm_rows merge_columns filter_keep filter_lose
merge_otus`

arch_plot **Base function** for generating a plot that compares the species abundance distribution for two sampled communities.

-b --basis

Defines the basis for removing samples in `rm_column` or `rm_rows`.

bdiv_metrics **Base function** that generates a table of standard biodiversity metrics from an OTU table.

dispersions **Base function** for measures of dispersal of data from an underlying distribution.

-fc --filter_column

Title of a column in an OTU table in `merge_columns`, `filter_keep` or `filter_lose`.

-fi --filter_items

Names used as a filter of an OTU table in `merge_columns` `merge_otus`, `filter_keep` or `filter_lose`.

filter_keep **Base function** that that uses items specified by **-fi** in one column specified by **-fc** to retain OTUs, removing all others.

filter_lose **Base function** that uses items specified by **-fi** in one column specified by **-fc** to remove OTUs from the dataset.

fit_SAD **Base function** that fits SAD models to OTU table data.

-gf --graph_format

Format for graphs in `quantify` and `fit_SAD`. Options = `svg png jpg`

-gs --graph_scale is a scaling factor for Venn diagrams.

-in--in_file

Name and path of a .csv OTU table for analysis.

merge_columns **Base function** for merging columns, which are renamed with **-nn** and defined by **-mt**.

merge_otus **Base function** for merging OTUs, which are renamed with **-nn** and defined by **-mt**.

merge_otus_auto **Base function** for merging OTUs based on common words in a column designated by **-fc**.

-me --metric

The metric to use for `quantify`. Options: `RRA RwRRA POO wPOO PRA`

-mfp --max_fit_plot

The maximum number of the best fitted SADs to plot with `fit_SAD`.

-minR --minimum_R

Minimum OTU richness for samples in an OTU table to be retained in `rm_columns`.

`-minReads --minimum_reads`
Minimum read counts for keeping samples in an OTU table in `rm_columns`.

`-mo --model`
The model to use for `quantify`. Options: `logser zipf genpareto lognorm geom linear nbinom`.

`-mt --merge_titles`
Name for columns or rows to be combined when merging in `merge_columns` or `merge_otus`.

`-nn --new_name`
Name for columns or rows formed with `merge_columns` or `merge_otus`.

`-od --out_dir`
Directory for file output.

`-of --out_dir`
Name basis for output files - file extensions and descriptions are added.

`-rn --rm_names`
If "names" is specified for `-basis`, the names of samples (columns) to remove from an OTU table in `rm_samples` and `rm_rows`.

`-pch --patchiness`
Proportion of OTUs generated lost due to expected eDNA patchiness in `sim_samples`.

`-pr --pseudoreplicates`
Number of pseudoreplicate datasets generated for estimating confidence in rank differences in `quantify`.

`-pt --POO_threshold`
Minimum POO for OTUs to be retained in `rm_rows`.

`-pv --p_value`
The p value for Brunner-Munzel tests for rank differences assessed on pseudoreplicated data in `fit_SAD`.

`-py --plot_yaxis`
Y axis choices for plots in `quantify` and `fit_SAD`. Options = `log linear`

Quantify Base function for estimating species abundance from a given SAD.

`-qt --quant_threshold`
Minimum estimated biomass proportion for an OTU to be retained in `rm_rows`.

`rm_columns` **Base function** for removing OTU table columns by a criterion defined by `--basis`.

`rm_rows` **Base function** for removing OTU table rows by a criterion defined by `--basis`.

`sim_samples` **Base function** for generating an OTU table from a simulated community.

-simC --sim_counts
Read counts for OTU table **sim_samples**.

-simN --sim_samples
Number of samples for OTU table simulated in **sim_samples**.

-simR --sim_richness
Species richness for communities pseudosampled in **sim_samples**.

-sp --shape_par
Shape metric for defining an SAD model for **quantify**, **sim_samples**.

-tr --transform
The data transform to use for **quantify**. Options: **Untransf**
arcsin_sqrt expit_lim clr

-ttr --test_transforms
A list of metrics to test SAD fit with using base command **fit_SAD**. Options:
Untransf arcsin_sqrt expit_lim

-tme --test_metric
A list of metrics to test SAD fit with using base command **fit_SAD**. Options:
RRA RwRRA POO wPOO PRA

-tmo --test_models
A list of models to test SAD fit with using base command **fit_SAD**.
Options: **nbinom logser zipf genpareto lognorm geom**

venn **Base function** for generating a Venn diagram that compares the shared and unique species incidence for two sampled communities.

5.0 References

- Aitchison, J. 1983. "Principal Component Analysis of Compositional Data." *Biometrika* 70 (1): 57–65.
- Akaike, H. 1974. "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* 19 (6): 716–23.
- Alberdi, Antton, and M. Thomas P. Gilbert. 2019. "A Guide to the Application of Hill Numbers to DNA-Based Diversity Analyses." *Molecular Ecology Resources* 19 (4): 804–17.
- Deagle, Bruce E., Austen C. Thomas, Julie C. McInnes, Laurence J. Clarke, Eero J. Vesterinen, Elizabeth L. Clare, Tyler R. Kartzinell, and J. Paige Eveson. 2019. "Counting with DNA in Metabarcoding Studies: How Should We Convert Sequence Reads to Dietary Data?" *Molecular Ecology* 28 (2): 391–406.
- Edgar, Robert C. 2010. "Search and Clustering Orders of Magnitude Faster than BLAST." *Bioinformatics* 26 (19): 2460–61.
- Fisher, R. A., A. Steven Corbet, and C. B. Williams. 1943. "The Relation between the Number of Species and the Number of Individuals in a Random Sample of an Animal Population." *The Journal of Animal Ecology* 12 (1): 42.
- Gloor, Gregory B., Jean M. Macklaim, Vera Pawlowsky-Glahn, and Juan J. Egozcue. 2017. "Microbiome Datasets Are Compositional: And This Is Not Optional." *Frontiers in Microbiology* 8 (November): 2224.
- Hurlbert, Stuart H. 1971. "The Nonconcept of Species Diversity: A Critique and Alternative Parameters." *Ecology* 52 (4): 577–86.
- Jarman, Simon, Jason B. Alexander, Kathryn L. Dawkins, Sherralee S. Lukehurst, Georgia M. Nester, Shaun Wilkinson, Michael J. Marnane, Justin I. McDonald, Travis S. Elsdon, and Euan S. Harvey. 2024. "Marine EDNA Sampling from Submerged Surfaces with Paint Rollers." *Marine Genomics* 76 (101127): 101127.
- Preston, F. W. 1948. "The Commonness, and Rarity, of Species." *Ecology* 29 (3): 254–83.
- Quinn, Thomas P., Ionas Erb, Mark F. Richardson, and Tamsyn M. Crowley. 2018. "Understanding Sequencing Data as Compositions: An Outlook and Review." *Bioinformatics (Oxford, England)* 34 (16): 2870–78.
- Rémond de Montmort, P. n.d. "Essai d'analyse Sur Les Jeux de Hasard." <http://sites.mathdoc.fr/cgi-bin/linum?aun=001130>.
- Rognes, Torbjørn, T. Flouri, Ben Nichols, C. Quince, and F. Mahé. 2016. "VSEARCH: A Versatile Open Source Tool for Metagenomics." *PeerJ* 4 (October). <https://doi.org/10.7717/peerj.2584>.
- Shannon, C. 1948. "A Mathematical Theory of Communication." *Bell Syst. Tech. J.* 27 (July): 623–56.
- Simpson, E. H. 1949. "Measurement of Diversity." *Nature* 163 (4148): 688–688.
- Zipf, G. K. 1935. "The Psycho-Biology of Language, Patterns."