# Morty v1.0

## **M**easurements **o**f Diver**s**i**ty**

August 17, 2020

## 0. Table of contents



## 1. Introduction

### 1.1. Overview

Morty is a Python library for measuring the *alpha* (intragroup) and/or *beta* (inter-group) diversity of a *metacommunity* composed of two constituent *subcommunities*. Morty was initially developed to handle the very large communities represented by a person's B- and T-cell repertoires (Arora and Arnaout, 2020), but can be used to measure alpha and beta diversity for any large or complex system.

### 1.2. Background and Terminology

*1.2.1. Diversity Indices*

At its core, diversity is simply a count: for example, a count of the number of unique species

present in a population. However, in many situations a more accurate or useful representation of the population can be had by selectively upweight the more common species relative to the rarer ones (in a sense, to be “less distracted” by rare species). Hill’s framework (Hill, 1973) makes it simple to control this weighting through the *viewpoint parameter, q*: the investigator can set $q$=0 to perform a count with no upweighting, or set $q$ to 0, 1, 2, …, ∞ (or any fractional value) to measure diversity with different weights. The resulting *diversity indices* are written ${}^{q}D$ and read "D-q" (e.g., ${}^{0}D$ is read as "D-zero"), and have simple and natural relationships to e.g. Shannon entropy ($q$=1), the Gini-Simpson index ($q$=2), the Berger-Parker index ($q$=∞), and others. These indices are also called *D numbers*, *Hill numbers*, or *effective numbers* (as in, for example, the "effective-number" form of Shannon entropy). The reader is referred to the Wikipedia entry on diversity indices for further discussion.

*1.2.2 Diversity with Similarity*

In addition to using $q$ to upweight higher-frequency species, it is often also useful to consider the *similarity* between pairs of species in measuring diversity. The idea here is that an ecosystem consisting of, say, 100 very different animal and plant species seems intuitively more diverse than an ecosystem that consists of 100 lichens; incorporating similarity allows *D*-number measures to reflect this intuition.

In the implementation here, this is done by constructing a similarity matrix, $\boldsymbol{Z}$, whose axes are the species and whose entries are the pairwise similarity between each pair of species, with 0 being completely unique and 1 being completely similar. Note that ignoring similarity is the same as setting $\boldsymbol{Z}$ to the identity matrix, $\boldsymbol{I}$ (i.e., ones on the diagonal and zero everywhere else). The result is a set of *D*-numbers for *similarity classes*, or *class diversity*, which we write ${}^{q}D_S$ (Leinster and Cobbold’s ${}^{q}D_Z$) instead of ${}^{q}D$, where the *S* subscript denotes "with similarity." When comparing to ${}^{q}D_S$, ${}^{q}D$ can be considered *species* or *raw diversity*. The reader is referred to the work of Leinster and Cobbold and of Reeve *et al.* for development and details of the mathematical framework that we implement here (Leinster and Cobbold, 2012; Reeve et al., 2014).

Especially when ignoring similarity, diversity measured on samples from complex systems can substantially underestimate the diversity in the overall system from which the sample was taken. Morty corrects for this error using Recon (software available separately on GitHub; see below). Therefore, installation of Recon is required in order to use Morty. Including similarity decreases the effective number, decreasing the likelihood of error, especially for larger samples. The reader is referred to Kaplinsky and Arnaout for more details on Recon (Kaplinsky and Arnaout, 2016).

*1.2.3 Subcommunities and Metacommunities*

A *community* is a collection of individuals of different species. Each species $i$ is present at a given frequency, $p_i$. Morty calculates diveristy of individual communities and between pairs of communities. The two communities are referred to as *subcommunities*, and the two subcommunities together are referred to as a *metacommunity*. (In principle one can compare diversity among any number of subcommunities, but currently Morty is written for pairs.)

One chooses the subcommunities and the metacommunity based on the question at hand. For example, in immunology, we often have immune repertoires from two different individuals and desire some measure of the diversity within each repertoire, and of the overlap between them. In this case, each repertoire is a subcommunity and the two subcommunities together form the

metacommunity, and our measures will be the alpha diversity of each subcommunity and the beta diversity between them. Note that alpha and beta diversity are independent of each other and therefore can be measured independently of each other (Jost, 2007).

There are different ways one can measure beta diversity (Chiu et al., 2014; Jost, 2007; Reeve et al., 2014). Following Reeve et al. 2014, Morty outputs how *representative* of the metacommunity each subcommunity is; if two subcommunities of equal size have nothing in common, then each constitutes half of the diversity of the metacommunity, and the normalized *representativeness* of each subcommunity for the metacommunity is 0.5. The normalized representativeness of subcommunity 1 for the metacommunity is written $\overline{\rho}_1$ ("rho-bar 1"), and that of subcommunity 2 is $\overline{\rho}_2$. Note that generally, $\overline{\rho}_1 \neq \overline{\rho}_2$. Morty also outputs their average, $\overline{R}$.

In addition, also following Reeve et al., Morty outputs $\overline{\beta}_1$ ("beta-bar 1") and $\overline{\beta}_2$, which are the reciprocals of $\overline{\rho}_1$ and $\overline{\rho}_2$. The average of $\overline{\beta}_1$ and $\overline{\beta}_2$ is $\overline{B}$. The $\overline{\beta}_i$ values are interpreted as the effective number of distinct subcommunities "like" subcommunity *i* that the metacommunity contains. In the example above in which $\overline{\rho}_1$=0.5, $\overline{\beta}_1$=2, meaning that the metacommunity effectively contains two subcommunities like subcommunity 1. (As another example, if the U.S. state of California is subcommunity 1 and the United States is the metacommunity, $\overline{\beta}_1$ would indicate effectively how many distinct California-equivalents the United States comprises.) Thus, $\overline{B}$ is the average effective number of distinct subcommunities present in the metacommunity.

### 1.3. Features

Morty was built to compare pairs of repertoires, communities composed of very large numbers of species (made up of different amino-acid sequences) (Arora and Arnaout, 2020). Because large numbers of species means a similarity matrix $\boldsymbol{Z}$ that is potentially too large to store or load into memory, Morty includes the ability to generate the similarity matrix $\boldsymbol{Z}$ on the fly for calculating both alpha and beta diversities, as described in 1.2.

### 1.4. Citing Morty

Please cite Arora, R., and Arnaout, R. (2020). Private Antibody Repertoires Are Public. BioRxiv 2020.06.18.159699.

## 2. Installation

### 2.1. Availability

Morty is publicly available on Github (https://github.com/ArnaoutLab/morty) subject to the terms in the license (Section 5).

### 2.2. Requirements

Morty requires:

- Python 3 (tested using Python version 3.8; earlier versions of Python 3 should work but have not been tested)
- The numpy, Cython and Levenshtein python libraries (which can be installed using standard methods, e.g. pip)

- simlib (written by us), which can be compiled from similib.pyx using setup.py (provided as part of this GitHub repository) using the command:
  ```
  python3 setup.py build_ext --inplace
  ```

In addition, measuring alpha diversity without similarity (***Z=I***) requires:

- recon_v3.0.py, which is available on GitHub (https://github.com/ArnaoutLab/Recon)
- The scipy python library (which can be installed using standard methods, e.g. pip)

### 2.3. Supported Platforms

Morty has been tested on Macintosh OS X versions Mojave (10.14.6) and Catalina (10.15.5).

### 2.4. Latest Version

As of this writing, the latest version of Morty is 1.0.

## 3. Operation

This section describes the various modes for running Morty. It can be run to generate alpha diversity for one community, or beta diversity of two subcommunities (which together constitute the metacommunity).

### 3.1 Alpha Diversity `(-mo, --mode alpha)`

#### *3.1.1. Description*

Given a subcommunity in a text file as an input, `-mo alpha` outputs alpha diversity values for this subcommunity. The output includes both class (${}^{q}D_{S}$) and species (${}^{q}D$) alpha-diversity values.

The user can provide their own pre-calculated similarity matrix, formatted in numpy format (.npy) or as a .csv file. Alternatively, the user can specify a python function to calculate similarity matrix on the fly using Morty (see §3.3.2 for details and §3.4.2.1 for example). If no similarity matrix or such function is provided, Morty will assume the species are protein sequences (strings written using the standard 20-amino-acid alphabet), and will calculate similarity using a multiplicatively independent model in which each single edit-distance difference between two species carries a cost *c* (recall the initial motivation behind Morty was calculating similarity in immune repertoires (Arora et al., 2018)).

#### *3.1.2. Usage*

```
python3 morty.py --mode alpha --input_files "filename.txt" --master_output_dir
output_path --recon_files "filename_for_recon.txt" --community_names
"subcommunity_name" [-qs [0.,1.,...] -v]
```

#### *3.1.3. Input*

Morty takes `filename.txt` as input, passed using the `-if/--input_files` command-line parameter. The filename must be enclosed between quotes. Each input file should consist of two columns, delimited by a tab character (`\t`). Rows must be delimited by a newline character

(`\n`). The first column must contain the name of the species (or some other species data), and the second column an integer count giving the frequency of that species. Thus, each row should have the form:

```
species\tcount\n
```

(Here, `\t` and `\n` denote the tab and newline characters, respectively.) Note, if a species appears multiple times in the file, the frequencies will be added, not overwritten.

The input file to get the Recon estimate `filename_for_recon.txt` can be specified using the command line option `-rf/--recon_files`. This may or may not be same as `filename.txt`.

*3.1.4. Output*

Calling Morty according to the command in §3.1.2 will append results to a master output file, specified by `-ma/--master_filename_alpha` which by default is `alpha_diversity_master_file.txt` and is found in `master_output_dir`. If no such file exists, one will be created.

Each time Morty is run in alpha mode, its output is appended to this master output file, with one new line for each subcommunity. This output contains information in the following order, with each piece of information separated by a tab (`\t`):

1. A unique `run_id` code that identifies the run (i.e., the specific instance of Morty's execution)
2. Type of diversity being calculated (`alpha` in the case of alpha diversity)
3. Name of the subcommunity for which the alpha diversity is calculated (taken from `filename` in the list of filenames following the `-if` parameter, if no repertoire names are provided using the `-cn/--community_names` parameter)
4. A tuple of two dictionaries (in standard Python input syntax), one for species diversities (${}^qD$) and one for class diversities (${}^qD_S$). The dictionary keys are the $q$ values and the dictionary values of are the corresponding ${}^qD$ (first dictionary) or ${}^qD_S$ (second dictionary) alpha-diversity values
5. Timestamp at which the code was run
6. A copy of the command that was run (for easy future reference)

**3.2. Beta Diversity (`-mo, --mode beta`)**

*3.2.1 Description*

For two subcommunities that constitute a metacommunity, `-mo/--mode beta` outputs the following beta diversity values: $\overline{\rho}$ (rho_bar), $\overline{\beta}$ (beta_bar), $\overline{R}$ (R_bar) and $\overline{B}$ (B_bar). Note that two different values of rho_bar and beta_bar will be calculated—$\overline{\rho}_1$ and $\overline{\rho}_2$, and $\overline{\beta}_1$ and $\overline{\beta}_2$—since there are two subcommunities being considered (see §1.2). For each of the above, values according to species (${}^qD$) and class diversity (${}^qD_S$) measures will be calculated, for the $q$ values indicated.

*3.2.2. Usage*

```
python3 morty.py --mode beta --input_files "filename1.txt,filename2.txt" --
master_output_dir output_path --community_names
"subcommunity1_name,subcommunity2_name" [--list_of_qs [0.,1.,...] --verbose]
```

*3.2.3. Input*

Input files and formatting are as in §3.1.3. The only difference is that beta diversity expects two input files and their corresponding community names.

*3.2.4. Output*

Calling Morty according to the command in §3.2.2 will append results to a master output file, specified by `-mb/--master_filename_beta` which by default is `beta_diversity_master_file.txt` and is found in `master_output_dir`. If no such file exists, one will be created.

Output is added to the output file line by line. Six new lines in total will be added for a single run in `beta` mode, with lines in the following order: B_bar, R_bar, beta_bar for subcommunity 1, beta_bar for subcommunity 2, rho_bar for subcommunity 1, and rho_bar for subcommunity 2. In turn, analogous to the output when run in alpha mode (§3.1.4), each line contains information in the following order, separated by a tab character:

1. A unique `run_id` code that identifies the run
2. Type of beta diversity (`B_bar`, `R_bar`, etc.)
3. Names of the subcommunities. For beta_bar and rho_bar, the resulting diversity values are measured from the perspective of the subcommunity that is mentioned first
4. A tuple of two dictionaries (in standard Python syntax), one for species diversities (${}^{q}D$) and one for class diversities (${}^{q}D_S$). The dictionary keys are the *q* values and the dictionary values of are the corresponding ${}^{q}D$ (first dictionary) or ${}^{q}D_S$ (second dictionary) alpha-diversity values
5. Timestamp at which the code was run
6. The command that was run (for easy reference)

**3.3. Run Parameters**

*3.3.1 Required parameters*

| | |
|---|---|
| `-mo, --mode` | Options are `alpha` or `beta` |
| `-if, --input_files` | One input file name/path for --mode alpha, and two input filenames/path for --mode beta where the filenames are separated by a comma. These files correspond to communities and contain species information and its count, in the format defined in §3.4.1. |
| `-cn, --community_names` | Name of the subcommunities, separated by a comma. |

| | |
|---|---|
| | Order should match `--input_files`. |
| `-mod, --master_output_dir` | User's local directory where the alpha and beta master output files will be written. |

*3.3.2. Optional parameters*

| | |
|---|---|
| `-qs, --list_of_qs` | List of *q* values for which diversity is to be calculated. The default is `[0., 0.5, 1., 1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6., 6.5, 7., 7.5, 8., 8.5, 9., 9.5, 10., inf]` Note the format: the *q*-values must be contained within square brackets (like a python list) and passed as strings (i.e. with quotes). |
| `-Z, --user_similarity_matrix_file` | A numpy (.npy format) or a comma-separated (.csv format) file that contains the user's pre-calculated similarity matrix. The user must supply a text file that contains the column/row header by using the `-uq/--unique_species_user` parameter (see examples in §3.4.2 for implementation). The user must ensure that the order of the species is identical in both the column and the row header such that the diagonal of the similarity matrix is the same. This is should be an all-against-all square matrix. Note that the default is `none` and in this default setting, Morty will calculate the similarity matrix (see §3.1.1). |
| `-uq, --unique_species_user` | A text file that contains the name of the species, separated by `\n`, in the order that is used for constructing a similarity matrix, using `-Z/--user_similarity_matrix_file`. The order of the species in this text file must be the order of the items in both the row and column dimensions. See §3.4.2 for examples. |
| `-ZF, --user_similarity_function` | A comma-separated string of length of 2 in the following order: (1) name of the user-defined python function that the to calculate all-against-all similarity, and (2) path to the .py file from which the python function will be imported. It is the user's responsibility to ensure that the python function must accept only the list of unique species as an input, which is the column and row header of the similarity matrix, and return only the complete similarity matrix. See §3.4.2 for examples. |

| | |
|---|---|
| | Note that the default is "get_similarity_matrix,path/to/get_fast_similarity.py" which has been optimized for immune repertoire amino acid sequences (see §3.1.1). |
| -v, --verbose | Turn on verbose output. |

### 3.4 Examples

This section will demonstrate usage of various command-line options on some use-cases of Morty using example data. Note that when Morty is run (irrespective of the `mode`), both types of diversity metric—with and without similarity (also referred to as "class" and "species/raw" diversity)—are calculated. The example commands for data generation and Morty execution can also be found in morty/example.ipynb

Note: Example data files generated here should exist in Example/ for a fresh download. Running this code will overwrite them.

*3.4.1 Run Morty using default similarity function.*

By default, Morty will use the similarity function written for amino-acid sequences used in Arora *et al.* 2018 and Arora & Arnaout 2020.

Generate community input files with data resembling amino-acid sequences:

```
with open("Examples/immune_test_1_species_to_count.txt", "w") as f,
open("Examples/immune_test_2_species_to_count.txt", "w") as g, open("Examples/
immune_test_3_species_to_count.txt", "w") as h:

    f.write("CARPQST\t1\n")
    f.write("CARPQSP\t1")
    g.write("CARPQST\t1\n")
    g.write("MALMNN\t1")
    h.write("WFALPCV\t1\n")
    h.write("EEIYREEEEE\t1")
```

Run alpha diversity on one of these communities:

Command:

```
python3 morty.py -cn 'immune_test_1' -mo alpha -if
'Examples/immune_test_1_species_to_count.txt' -qs '[0., 1., 2., inf]' -rf
'Examples/immune_test_1_species_to_count.txt' --master_output_dir 'Examples'
```

Screen output:

```
2020-08-14 07:50:13

run_id: TptRd

Similarity matrix will be generated by:
Function:       get_similarity_matrix
```

```
Imported from:
/Users/rohitarora/Documents/GitHub/morty/get_fast_similarity.py
Output for run TptRd written to:
Examples/alpha_diversity_master_file.txt

2020-08-14 07:50:14
```

The results of this run can either be viewed in real time by using `--verbose` option when running the command, or looking for the run_id `tmO6T` in `Examples/alpha_diversity_master_file.txt`. For a quick look, simply use `grep`.

Command:
```
grep  TptRd Examples/alpha_diversity_master_file.txt
```

Screen output:
```
TptRd   alpha   immune_test_1   ({'0.0Ds': '1.325e+00', '1.0Ds':
'1.325e+00', '2.0Ds': '1.325e+00', 'infDs': '1.325e+00'}, {'0.0D':
'8.100e+01', '1.0D': '8.100e+01', '2.0D': '8.100e+01', 'infD':
'8.100e+01'})        2020-08-14 07:50:13     python morty.py -cn
immune_test_1 -mo alpha -if
Examples/immune_test_1_species_to_count.txt -qs [0., 1., 2., inf] -rf
Examples/immune_test_1_species_to_count.txt --master_output_dir
Examples
```

In this community, the two species look very similar (per this similarity metric), and therefore the effective diversity with similarity is closer to 1. This value is closer to 2, as the species are more dissimilar. User is encouraged to repeat this example with the remaining two example community data files.

Now, we run beta diversity on pairs of these communities.

Command:
```
python3 morty.py -cn 'immune_test_1,immune_test_2' -mo beta -if
"Examples/immune_test_1_species_to_count.txt,Examples/immune_test_2_species_to
_count.txt" -qs '[0.0]' --verbose --master_output_dir 'Examples'
```

Screen output:
```
2020-08-14 08:08:12

run_id: BmIWe

Similarity matrix will be generated by:
Function:       get_similarity_matrix
Imported from:
/Users/rohitarora/Documents/GitHub/morty/get_fast_similarity.py
Output for run BmIWe written to:
Examples/beta_diversity_master_file.txt

2020-08-14 08:08:12
```

As discussed above, results of this run can be viewed in real time by using the `--verbose` option in the command or by using grep:

Command:

```
grep BmIWe Examples/beta_diversity_master_file.txt
```

Screen output:

```
BmIWe   B_bar   immune_test_1   immune_test_2   {'0.0Ds': '1.230e+00', '0.0D':
'1.333e+00'}     2020-08-14 08:08:12     python morty.py -cn
immune_test_1,immune_test_2 -mo beta -if
Examples/immune_test_1_species_to_count.txt,Examples/immune_test_2_species_to_
count.txt -qs [0.0] --master_output_dir Examples

BmIWe   R_bar   immune_test_1   immune_test_2   {'0.0Ds': '8.177e-01', '0.0D':
'7.500e-01'}     2020-08-14 08:08:12     python morty.py -cn
immune_test_1,immune_test_2 -mo beta -if
Examples/immune_test_1_species_to_count.txt,Examples/immune_test_2_species_to_
count.txt -qs [0.0] --master_output_dir Examples

BmIWe   beta_bar        immune_test_1   immune_test_2   {'0.0Ds': '1.323e+00',
'0.0D': '1.333e+00'}     2020-08-14 08:08:12     python morty.py -cn
immune_test_1,immune_test_2 -mo beta -if
Examples/immune_test_1_species_to_count.txt,Examples/immune_test_2_species_to_
count.txt -qs [0.0] --master_output_dir Examples

BmIWe   beta_bar        immune_test_2   immune_test_1   {'0.0Ds': '1.137e+00',
'0.0D': '1.333e+00'}     2020-08-14 08:08:12     python morty.py -cn
immune_test_1,immune_test_2 -mo beta -if
Examples/immune_test_1_species_to_count.txt,Examples/immune_test_2_species_to_
count.txt -qs [0.0] --master_output_dir Examples

BmIWe   rho_bar immune_test_1   immune_test_2   {'0.0Ds': '7.558e-01', '0.0D':
'7.500e-01'}     2020-08-14 08:08:12     python morty.py -cn
immune_test_1,immune_test_2 -mo beta -if
Examples/immune_test_1_species_to_count.txt,Examples/immune_test_2_species_to_
count.txt -qs [0.0] --master_output_dir Examples

BmIWe   rho_bar immune_test_2   immune_test_1   {'0.0Ds': '8.796e-01', '0.0D':
'7.500e-01'}     2020-08-14 08:08:12     python morty.py -cn
immune_test_1,immune_test_2 -mo beta -if
Examples/immune_test_1_species_to_count.txt,Examples/immune_test_2_species_to_
count.txt -qs [0.0] --master_output_dir Examples
```

Every beta run generates 6 entries in the output file, each corresponding to one of the beta diversity parameters, as discussed in §3.2.1. The parameter that corresponds to overlap is `R_bar`. In this run, the two communities share a species, and therefore `R_bar` with similarity is relatively high. We expect this parameter to be lower if we measure beta diversity when two disjoint communities are compared, as we show in the following run. `R_bar` is closer to the minimum value of 0.5 (`min(R_bar)` is $1/n$ where $n$ is the number of communities; here $n=2$):

Command:

```
python3 morty.py -cn 'immune_test_1,immune_test_3' -mo beta -if
"Examples/immune_test_1_species_to_count.txt,Examples/immune_test_3_species_to
_count.txt" -qs '[0.0]' --master_output_dir 'Examples'
```

Screen output:

```
2020-08-14 08:31:45
```

```
run_id: BoxSY

Similarity matrix will be generated by:
Function:       get_similarity_matrix
Imported from:
/Users/rohitarora/Documents/GitHub/morty/get_fast_similarity.py
Output for run BoxSY written to:
Examples/beta_diversity_master_file.txt

2020-08-14 08:31:45
```

Access the `R_bar` entry for this run.

Command:

```
grep BoxSY Examples/beta_diversity_master_file.txt | grep R_bar
```

Screen output:

```
BoxSY   R_bar   immune_test_1   immune_test_3   {'0.0Ds': '5.083e-01', '0.0D':
'5.000e-01'}     2020-08-14 08:31:45     python morty.py -cn
immune_test_1,immune_test_3 -mo beta -if
Examples/immune_test_1_species_to_count.txt,Examples/immune_test_3_species_to_
count.txt -qs [0.0] --master_output_dir Examples
```

### *3.4.2 Run Morty with a custom similarity matrix or function*

Since the default similarity function is optimized for immune repertoires, which may not be useful for other systems, Morty allows users to either (*i*) pre-calculate a similarity matrix and supply it to Morty, or (*ii*) point Morty to a python function which can be used to calculate similarity on the fly. We demonstrate use of these options here on alpha diversity calculations in a non-immune repertoire system.

Generate data for a community from the iris dataset.

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris()
iris_data = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
                     columns= iris['feature_names'] + ['target'])
iris_input = iris_data[['sepal length (cm)', 'sepal width (cm)']]
iris_input.to_csv('Examples/iris_input.csv')

# Process input csv to generate community input files, and files with unique
species which defines the species order for the similarity matrix.
def process_input(iris_csv_file, species_to_count_file, unique_species_file):
      processed_input_list=[]
      with open(iris_csv_file) as f, open(species_to_count_file, "w") as g,
open(unique_species_file, "w") as h:
            for line in f:
                  if "sepal" in line: continue
                  id_, sepal_length, sepal_width = str.strip(line).split(",")
                  out_str = "%s_%s_%s" % (id_, sepal_length, sepal_width)
                  processed_input_list.append((out_str))
```

```
                g.write("%s\t1\n" % id_)
                h.write("%s\n" % id_)
        return processed_input_list

iris_input_list = process_input('Examples/iris_input.csv',
'Examples/iris_input_species_to_count.txt',
'Examples/iris_input_uniq_species.txt')
```

In the context of this example, we define similarity between two species as the euclidean distance between sepal length and width, and write an example function to that end.

```
from scipy.spatial import distance

def get_euclidean_distance(species_list):
        """This function calculates all-against-all euclidean distance
        between 2-D coordinates of two species"""

        species_list = [ i.split("_") for i in species_list ]

        similarity_list=[]
        for species_1, x1, y1 in species_list:
                similarity_for_this_species=[]

                for species_2, x2, y2 in species_list:
                        if species_1 == species_2: sim_ = 1.
                        else: sim_ = distance.euclidean((float(x1),float(y1)),
(float(x2), float(y2)))
                        similarity_for_this_species.append(sim_)
                similarity_list.append(similarity_for_this_species)
        similarity_matrix = np.array(similarity_list)
        return  similarity_matrix
```

We write this function to `Examples/euclidean_similarity.py` to access it later.

*3.4.2.1. User-entered similarity matrix*

Now, we calculate the similarity matrix for this iris community outside of Morty and then supply this matrix to Morty.

```
similarity_matrix_input = get_euclidean_distance(iris_input_list)
np.save("Examples/similarity_matrix_input.npy", similarity_matrix_input)
```

Command:

```
python3 morty.py -cn "iris_input" -mo alpha -if
"Examples/iris_input_species_to_count.txt" -qs "[0., 1., 2., inf]" -rf
"Examples/iris_input_species_to_count.txt" -Z
"Examples/similarity_matrix_input.npy" -uq
"Examples/iris_input_uniq_species.txt" --master_output_dir "Examples"
```

Screen output:

```
2020-08-14 10:43:54

run_id: YYphl
```

```
Note: Using user-generated similarity matrix.

You have chosen to use a user-generated similarity matrix.
User-generated similarity matrix file:  Examples/similarity_matrix_input.npy
User-generated unique species order file:
Examples/iris_input_uniq_species.txt
Output for run YYphl written to:
Examples/alpha_diversity_master_file.txt

2020-08-14 10:43:55
```

Access the results in the output file.

Command:

```
grep YYphl Examples/alpha_diversity_master_file.txt
```

Screen output:

```
YYphl   alpha   iris_input      ({'0.0Ds': '9.070e-01', '1.0Ds': '8.883e-01',
'2.0Ds': '8.677e-01', 'infDs': '4.422e-01'}, {'0.0D': '6.075e+03', '1.0D':
'6.075e+03', '2.0D': '6.075e+03', 'infD': '6.075e+03'})        2020-08-14
10:43:54     python morty.py -cn iris_input -mo alpha -if
Examples/iris_input_species_to_count.txt -qs [0., 1., 2., inf] -rf
Examples/iris_input_species_to_count.txt -Z
Examples/similarity_matrix_input.npy -uq Examples/iris_input_uniq_species.txt
--master_output_dir Examples
```

*3.4.2.2. Calculating similarity matrix on the fly (user-defined similarity function)*

Now we repeat this calculation, but instead of supplying Morty with the similarity matrix, we point Morty to the function and let it calculate this matrix in real time.

Command:

```
python3 morty.py -cn "iris_input" -mo alpha -if
"Examples/iris_input_species_to_count.txt" -qs "[0., 1., 2., inf]" -rf
"Examples/iris_input_species_to_count.txt" -ZF
"get_euclidean_distance,Examples/euclidean_similarity.py" --master_output_dir
"Examples"
```

screen output:

```
2020-08-14 11:12:32

run_id: h3xTD

Similarity matrix will be generated by:
Function:       get_euclidean_distance
Imported from:  Examples/euclidean_similarity.py
Output for run h3xTD written to:
Examples/alpha_diversity_master_file.txt

2020-08-14 11:12:34
```

Command:

```
grep h3xTD Examples/alpha_diversity_master_file.txt
```

Screen output:

```
h3xTD   alpha   iris_input      ({'0.0Ds': '9.070e-01', '1.0Ds': '8.883e-01',
'2.0Ds': '8.677e-01', 'infDs': '4.422e-01'}, {'0.0D': '6.075e+03', '1.0D':
'6.075e+03', '2.0D': '6.075e+03', 'infD': '6.075e+03'})        2020-08-14
11:12:32     python morty.py -cn iris_input -mo alpha -if
Examples/iris_input_species_to_count.txt -qs [0., 1., 2., inf] -rf
Examples/iris_input_species_to_count.txt -ZF
get_euclidean_distance,Examples/euclidean_similarity.py --master_output_dir
Examples
```

Note that the alpha diversity results from `YYphl` and `h3xTD` are identical. That is what we should expect since the measurment is on the same data, the only difference being that in the former case, we calculated the similarity matrix outside of Morty. This gives the user the flexibility to evaluate the similarity matrix with a function of their choice, even if the function is not compatible with Morty. In the latter case, we pointed Morty to the same function, since it is already compatible with Morty.

### 3.5. Unit Testing (`-u, --unit_test`)

*3.5.1 Description*

Performs internal checks to ensure that the script runs without error.

*3.5.2 Usage*

```
python3 morty.py -u
```

*3.5.3 Output*

The above command line will test alpha class diversity, beta class diversity, and beta species diversity for q = [0.0, 1.0, 2.0, 3.0, 3.5] and print out the results ("`pass`" or otherwise warning signs) to standard output. To test species alpha diversity, the unit test of recon_v3.0.py needs to be run (not run by Morty).

## 4. Contact Information

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

## 5. License

## 6. References

Arora, R., and Arnaout, R. (2020). Private Antibody Repertoires Are Public. BioRxiv 2020.06.18.159699.

Chiu, C.-H., Jost, L., and Chao, A. (2014). Phylogenetic beta diversity, similarity, and differentiation measures based on Hill numbers. Ecological Monographs *84*, 21–44.

Hill, M.O. (1973). Diversity and Evenness: A Unifying Notation and Its Consequences. Ecology *54*, 427–432.

Jost, L. (2007). Partitioning diversity into independent alpha and beta components. Ecology *88*, 2427–2439.

Kaplinsky, J., and Arnaout, R. (2016). Robust estimates of overall immune-repertoire diversity from high-throughput measurements on samples. Nat Commun *7*, 11881.

Leinster, T., and Cobbold, C.A. (2012). Measuring diversity: the importance of species similarity. Ecology *93*, 477–489.

Reeve, R., Leinster, T., Cobbold, C.A., Thompson, J., Brummitt, N., Mitchell, S.N., and Matthews, L. (2014). How to partition diversity. ArXiv:1404.6520 [q-Bio].