sourcepredict Documentation

Release 0.3.2

Maxime Borry

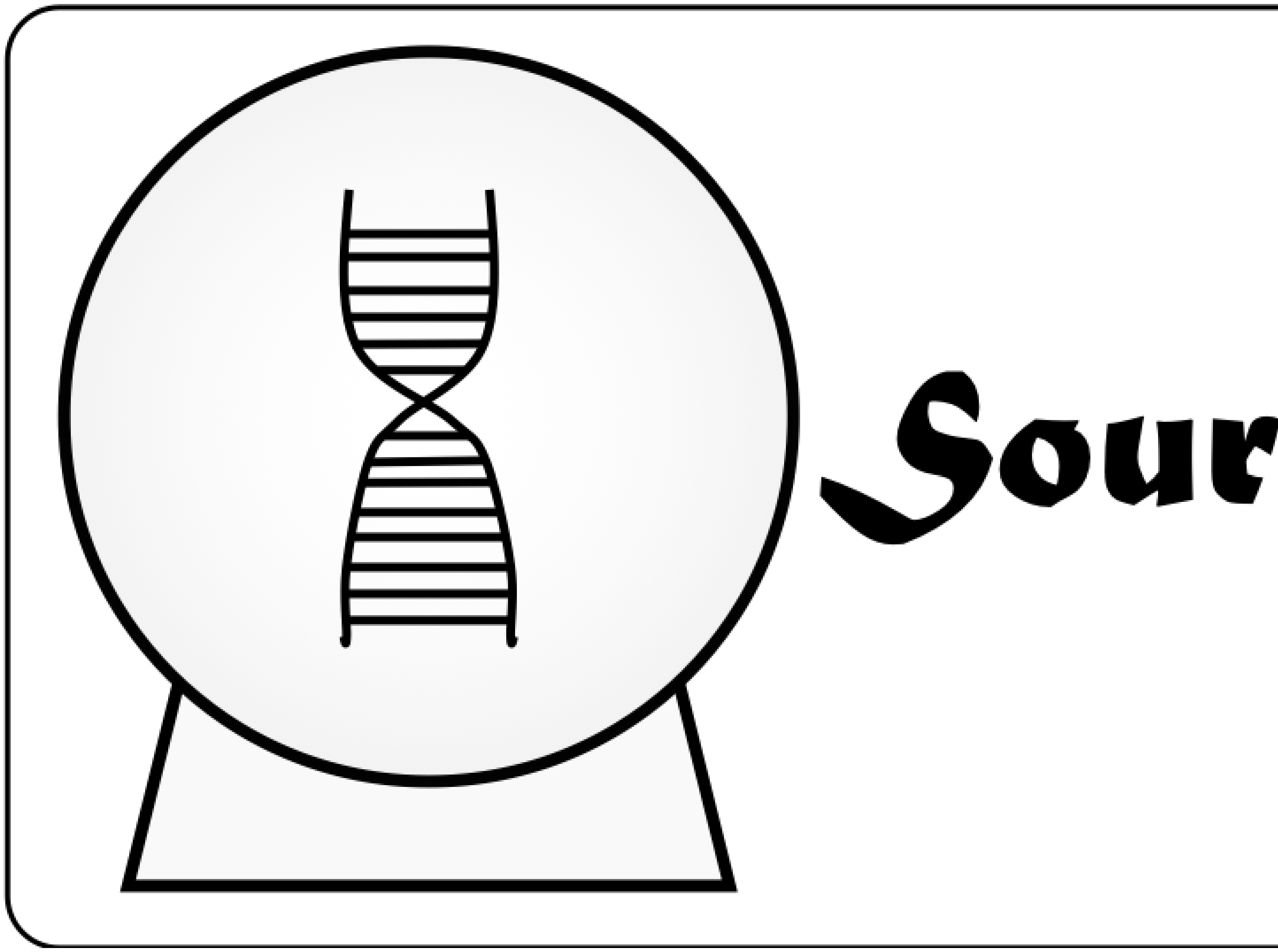
Apr 15, 2020

Contents:

Homepage: github.com/maxibor/sourcepredict

CHAPTER 1

Introduction



Prediction/source tracking of metagenomic samples source using machine learning

---

SourcePredict (github.com/maxibor/sourcepredict) is a Python package distributed through Conda, to classify and predict the origin of metagenomic samples, given a reference dataset of known origins, a problem also known as source tracking.DNA shotgun sequencing of human, animal, and environmental samples has opened up new doors to explore the diversity of life in these different environments, a field known as metagenomics.One aspect of metagenomics is investigating the community composition of organisms within a sequencing sample with tools known as taxonomic classifiers, such as Kraken.

In cases where the origin of a metagenomic sample, its source, is unknown, it is often part of the research question to predict and/or confirm the source. For example, in microbial archaelogy, it is sometimes necessary to rely on metagenomics to validate the source of paleofaeces. Using samples of known sources, a reference dataset can be established with the taxonomic composition of the samples, *i.e.* the organisms identified in the samples as features, and the sources of the samples as class labels. With this reference dataset, a machine learning algorithm can be trained to predict the source of unknown samples (sinks) from their taxonomic composition.Other tools used to perform the prediction of a sample source already exist, such as SourceTracker sourcetracker, which employs Gibbs sampling.However, the Sourcepredict results are easier interpreted since the samples are embedded in a human observable low-dimensional space. This embedding is performed by a dimension reduction algorithm followed by K-Nearest-Neighbours (KNN) classification.

CHAPTER 2

Installation

SourcePredict can be installed using the conda package manager:

```
$ conda install -c conda-forge -c maxibor sourcepredict
```

CHAPTER 3

Usage

3.1 Running sourcepredict on the test dataset

```
$ wget https://raw.githubusercontent.com/maxibor/sourcepredict/master/data/test/dog_
↪test_sink_sample.csv -O dog_example.csv
$ wget https://raw.githubusercontent.com/maxibor/sourcepredict/master/data/modern_gut_
↪microbiomes_labels.csv -O sp_labels.csv
$ wget https://raw.githubusercontent.com/maxibor/sourcepredict/master/data/modern_gut_
↪microbiomes_sources.csv -O sp_sources.csv
$ sourcepredict -s sp_sources.csv -l sp_labels.csv dog_example.csv
```

3.2 Command line interface

```
$ sourcepredict -h
usage: SourcePredict v0.4 [-h] [-a ALPHA] [-s SOURCES] [-l LABELS]
                          [-n NORMALIZATION] [-dt DISTANCE] [-r TAX_RANK]
                          [-me METHOD] [-kne NEIGHBORS] [-kw WEIGHTS]
                          [-e EMBED] [-di DIM] [-o OUTPUT] [-se SEED]
                          [-k KFOLD] [-t THREADS]
                          sink_table

==========================================================
SourcePredict v0.4
Coprolite source classification
Author: Maxime Borry
Contact: <borry[at]shh.mpg.de>
Homepage & Documentation: github.com/maxibor/sourcepredict
==========================================================


positional arguments:
```

(continues on next page)

(continued from previous page)

```
  sink_table          path to sink TAXID count table in csv format

optional arguments:
  -h, --help          show this help message and exit
  -a ALPHA            Proportion of sink sample in unknown. Default = 0.1
  -s SOURCES          Path to source csv file. Default =
                      data/modern_gut_microbiomes_sources.csv
  -l LABELS           Path to labels csv file. Default =
                      data/modern_gut_microbiomes_labels.csv
  -n NORMALIZATION    Normalization method (RLE | Subsample | GMPR | None).
                      Default = GMPR
  -dt DISTANCE        Distance method. (unweighted_unifrac | weighted_unifrac)
                      Default = weighted_unifrac
  -r TAX_RANK         Taxonomic rank to use for Unifrac distances. Default =
                      species
  -me METHOD          Embedding Method. TSNE, MDS, or UMAP. Default = TSNE
  -kne NEIGHBORS      Numbers of neigbors if KNN ML classication (integer or
                      'all'). Default = 0 (chosen by CV)
  -kw WEIGHTS         Sample weight function for KNN prediction (distance |
                      uniform). Default = distance.
  -e EMBED            Output embedding csv file. Default = None
  -di DIM             Number of dimensions to retain for dimension reduction.
                      Default = 2
  -o OUTPUT           Output file basename. Default =
                      <sample_basename>.sourcepredict.csv
  -se SEED            Seed for random generator. Default = 42
  -k KFOLD            Number of fold for K-fold cross validation in parameter
                      optimization. Default = 5
  -t THREADS          Number of threads for parallel processing. Default = 2
```

3.3 Command line arguments

3.3.1 sink_table

Sink TAXID count table in `csv` file format

Example sink count table file

```
+-------+----------+----------+
| TAXID |  SINK_1  |  SINK_2  |
+-------+----------+----------+
|  283  |    5     |    2     |
+-------+----------+----------+
|  143  |     25   |    48    |
+-------+----------+----------+
```

3.3.2 -alpha

Proportion of alpha of sink sample in unknown. Default = `0.1` $$\alpha \in [0,1]$$

Example:

`-alpha 0.1`

3.3.3 -s SOURCES

Path to source `csv` (training) file with samples in columns, and TAXIDs in rows. Default = `data/sourcepredict/modern_gut_microbiomes_sources.csv`

Example:

`-s data/sourcepredict/modern_gut_microbiomes_sources.csv`

Example source file :

```
+-------+----------+----------+
| TAXID | SAMPLE_1 | SAMPLE_2 |
+-------+----------+----------+
|  467  |    18    |    24    |
+-------+----------+----------+
|  786  |     3    |    90    |
+-------+----------+----------+
```

3.3.4 -l LABELS

Path to labels `csv` file of sources. Default = `data/modern_gut_microbiomes_labels.csv`

Example:

`-l data/modern_gut_microbiomes_labels.csv`

Example source file :

```
+----------+--------+
|          | labels |
+----------+--------+
| SAMPLE_1 |   Dog  |
+----------+--------+
| SAMPLE_2 |  Human |
+----------+--------+
```

3.3.5 -n NORMALIZATION

Normalization method. One of `RLE`, `CLR`, `Subsample`, or `GMPR`. Default = `GMPR`

3.3.6 -dt DISTANCE

Distance method. One of `unweighted_unifrac`, `weighted_unifrac`. Default = `weighted_unifrac`

Example:

`-dt weighted_unifrac`

3.3.7 -me METHOD

Embedding Method. One of `MDS`, `TSNE` or `UMAP`. Default = `TSNE`

Example:

`-me TSNE`

3.3.8 -kne NEIGHBORS

Numbers of neigbors for KNN classication. Default = 0 (chosen by CV). Either an integer, or 'all'

Example:

`-kne 30` `-kne all`

> Setting the number of neighbors to 0 will let Sourcepredict choose the optimal number of neighbors for **classification**. If set to `all`, the KNN algorithm will use all the training samples. For **source proportion estimation**, setting `-kne` to *all* will give better estimations.See example 2 for illustration.

3.3.9 –kw WEIGHTS

Sample weight function for KNN prediction (distance | uniform). Default = distance.

Choose to give a uniform or distance based weights to neighbor samples in KNN algorithm.

> Distance base weights will work better for **classification** while uniform weigths will work better for **source proportion estimation**.See example 2 for illustration.

3.3.10 -e EMBED

File for saving embedding coordinates in `csv` format. Default = `None`

Example:

`-e embed_coord.csv`

3.3.11 -di DIM

Number of dimensions to retain for dimension reduction. Default = `2`

Example:

`-di 2`

3.3.12 -o OUTPUT

Sourcepredict Output file basename. Default = `<sample_basename>.sourcepredict.csv`

Example:

`-o my_output`

3.3.13 -se SEED

Seed for random number generator. Default = `42`

Example:

`-se 42`

3.3.14 -k KFOLD

Number of fold for K-fold cross validation in parameter optimization. Default = `5`

Example:

```
-k 5
```

3.3.15 -t THREADS

Number of threads for parallel processing. Default = `2`

Example:

```
-t 2
```

3.4 Choice of the taxonomic classifier

Different taxonomic classifiers will give different results, because of different algorithms, and different databases.

In order to produce correct results with Sourcepredict, **the taxonomic classifier and the database used to produce the *source* TAXID count table must be the same as the one used to produce the *sink* TAXID count table**.

Because Sourcepredict relies on machine learning, at least 10 samples per sources are required, but more source samples will lead to a better prediction by Sourcepredict.

Therefore, running all these samples through a taxonomic classifier ahead of Sourcepredict requires a non-negligeable computational time.

Hence the choice of the taxonomic classifier is a balance between precision, and computational time.

While this documentation doesn't intent to be a benchmark of taxonomic classifiers, the author of Sourcepredict has had decent results with Kraken2 and recommends it for its good compromise between precision and runtime.

The example *source* and *sink* data provided with Sourcepredict were generated with Kraken2.

> If you already have kraken report formatted results (from Kraken, KrakenUniq, Kraken2, Centrifuge, . . .), you can use the kraken_parse.py script to convert a kraken report to a column of a TAXID count table.

CHAPTER 4

Output

4.1 SourcePredict result file

File: `*.sourcepredict.csv`

This `csv` file contains the predicted proportion of each source in each sample. Like in any classification problem, the predicted source is the greatest proportion.

Example:

```
+-------------------+----------------------+
|                   |      ERR1915662      |
+-------------------+----------------------+
| Canis_familiaris  |  0.9449678590674971  |
+-------------------+----------------------+
|   Homo_sapiens    | 0.027033026106258438 |
+-------------------+----------------------+
|       Soil        | 0.014110223165444446 |
+-------------------+----------------------+
|      unknown      | 0.013888891660799834 |
+-------------------+----------------------+
```

While in this example it is pretty clear that the `ERR1915662` sample is likely a dog, you may face situations where it will be less obvious. Looking at the embedding can therefore be useful to decide from which source(s) the sink sample is made up of.

4.2 Embedding csv file

This `csv` file contains the embedding of training in test samples in lower dimensions by TSNE or UMAP

Example:

```
+------------------+------------+-------------+--------------+------------------+
|                  | PC1        | PC2         | labels       | name             |
+------------------+------------+-------------+--------------+------------------+
| SRR1175007       | -28.858526 | 0.59231776  | Homo_sapiens | SRR1175007       |
+------------------+------------+-------------+--------------+------------------+
| SRR042182        | -22.14415  | -0.47057405 | Homo_sapiens | SRR042182        |
+------------------+------------+-------------+--------------+------------------+
| SRR061154        | -30.210106 | -2.0323594  | Homo_sapiens | SRR061154        |
+------------------+------------+-------------+--------------+------------------+
| SRR061499        | -25.546652 | 0.27987793  | Homo_sapiens | SRR061499        |
+------------------+------------+-------------+--------------+------------------+
| SRR063469        | -22.88011  | 1.1526666   | Homo_sapiens | SRR063469        |
+------------------+------------+-------------+--------------+------------------+
| SRR062324        | -25.50832  | -0.25076494 | Homo_sapiens | SRR062324        |
+------------------+------------+-------------+--------------+------------------+
| SRR1179037       | -28.779644 | 0.1385772   | Homo_sapiens | SRR1179037       |
+------------------+------------+-------------+--------------+------------------+
| SRR061236        | -29.470839 | -0.8973783  | Homo_sapiens | SRR061236        |
+------------------+------------+-------------+--------------+------------------+
| SRR061456        | -28.31991  | -0.9834692  | Homo_sapiens | SRR061456        |
+------------------+------------+-------------+--------------+------------------+
| SRR1761669       | 4.1411834  | 14.485897   | Homo_sapiens | SRR1761669       |
+------------------+------------+-------------+--------------+------------------+
| SRR1761668       | 1.7706155  | 13.6566925  | Homo_sapiens | SRR1761668       |
+------------------+------------+-------------+--------------+------------------+
| SRR1761675       | 3.2434833  | 16.020077   | Homo_sapiens | SRR1761675       |
+------------------+------------+-------------+--------------+------------------+
| SRR3578625       | 24.127249  | 17.996181   | Soil         | SRR3578625       |
+------------------+------------+-------------+--------------+------------------+
| ERR1939165       | 28.738718  | 19.882471   | Soil         | ERR1939165       |
+------------------+------------+-------------+--------------+------------------+
| SRR3578645       | 24.138885  | 17.998867   | Soil         | SRR3578645       |
+------------------+------------+-------------+--------------+------------------+
| ERR1915662       | -14.770308 | -30.94284   | sink         | ERR1915662       |
+------------------+------------+-------------+--------------+------------------+
```

See the example usage of Sourcepredict for a example of how to plot it.

CHAPTER 5

Methods

Starting with a numerical organism count matrix (samples as columns, organisms as rows, obtained by a taxonomic classifier) of merged references and sinks datasets, samples are first normalized relative to each other, to correct for uneven sequencing depth using the GMPR method (default). After normalization, Sourcepredict performs a two-step prediction algorithm. First, it predicts the proportion of unknown sources, *i.e.* which are not represented in the reference dataset. Second it predicts the proportion of each known source of the reference dataset in the sink samples.

Organisms are represented by their taxonomic identifiers (TAXID).

5.1 Prediction of unknown sources proportion

Let $S_i \in \{S_1, .., S_n\}$ be a sample from the normalized sinks dataset D_{sink}, $o_j^{\ i} \in \{o_1^{\ i}, .., o_{n_o^{\ i}}^{\ i}\}$ be an organism in S_i, and $n_o^{\ i}$ be the total number of organisms in S_i, with $o_j^{\ i} \in \mathbb{Z}+$.

Let m be the mean number of samples per class in the reference dataset, such that $m = \frac{1}{O}\sum_{i=1}^{O} S_i$.

For each S_i sample, I define $||m||$ estimated samples $U_k^{S_i} \in \{U_1^{S_i}, .., U_{||m||}^{S_i}\}$ to add to the reference dataset to account for the unknown source proportion in a test sample.

Separately for each S_i, a proportion denoted $\alpha \in [0, 1]$ (default = 0.1) of each of the $o_j^{\ i}$ organism of S_i is added to each $U_k^{S_i}$ samples such that $U_k^{S_i}(o_j^{\ i}) = \alpha \cdot x_{i\ j}$, where $x_{i\ j}$ is sampled from a Gaussian distribution $\mathcal{N}\big(S_i(o_j^{\ i}), 0.01\big)$.

The $||m||$ $U_k^{S_i}$ samples are then added to the reference dataset D_{ref}, and labeled as *unknown*, to create a new reference dataset denoted ${}^{unk}D_{ref}$.

To predict the proportion of unknown sources, a Bray-Curtis pairwise dissimilarity matrix of all S_i and $U_k^{S_i}$ samples is computed using scikit-bio. This distance matrix is then embedded in two dimensions (default) with the scikit-bio implementation of PCoA.

This sample embedding is divided into three subsets: ${}^{unk}D_{train}$ (64%), ${}^{unk}D_{test}$ (20%), and ${}^{unk}D_{validation}$(16%).

The scikit-learn implementation of KNN algorithm is then trained on ${}^{unk}D_{train}$, and the training accuracy is computed with ${}^{unk}D_{test}$.
This trained KNN model is then corrected for probability estimation of the unknown proportion using the scikit-learn implementation of Platt's scaling method with ${}^{unk}D_{validation}$.

The proportion of unknown sources in S_i, $p_u \in [0, 1]$ is then estimated using this trained and corrected KNN model.

Ultimately, this process is repeated independantly for each sink sample S_i of D_{sink}.

5.2 Prediction of known source proportion

First, only organism TAXIDs corresponding to the species taxonomic level are retained using the ETE toolkit. A weighted Unifrac (default) pairwise distance matrix is then computed on the merged and normalized training dataset D_{ref} and test dataset D_{sink} with scikit-bio.

This distance matrix is then embedded in two dimensions (default) using the scikit-learn implementation of t-SNE.

The 2-dimensional embedding is then split back to training ${}^{tsne}D_{ref}$ and testing dataset ${}^{tsne}D_{sink}$.

The training dataset ${}^{tsne}D_{ref}$ is further divided into three subsets: ${}^{tsne}D_{train}$ (64%), ${}^{tsne}D_{test}$ (20%), and ${}^{tsne}D_{validation}$ (16%).
The KNN algorithm is then trained on the train subset, with a five (default) cross validation to look for the optimum number of K-neighbors. Finally, the training accuracy is then computed with ${}^{tsne}D_{test}$.

The proportion $p_{c_s} \in [0, 1]$ of each of the n_s sources $c_s \in \{c_1, \ .., \ c_{n_s}\}$ in each sample S_i is then estimated using this second trained and corrected KNN model.

5.3 Combining unknown and source proportion

Then for each sample S_i of the test dataset D_{sink}, the predicted unknown proportion p_u is then combined with the predicted proportion p_{c_s} for each of the n_s sources c_s of the training dataset such that $\sum_{c_s=1}^{n_s} s_c + p_u = 1$ where $s_c = p_{c_s} \cdot p_u$.

Finally, a summary table gathering the estimated sources proportions is returned as a `csv` file, as well as the t-SNE embedding sample coordinates.

CHAPTER 6

Custom sources

Different taxonomic classifers will give different results, and **the taxonomic classifier used to produce the *source* TAXID count table must be the same as the one used to produce the *sink* TAXID count table**.

While there are many available taxonomic classifiers available to produce the source and sink TAXID table, the Sourcepredict author provide a simple pipeline to generate the source and sink TAXID table.

This pipeline is written using Nextflow, and handles the dependancies using conda. Briefly, this pipelines will firt trim and clip the sequencing files with AdapterRemoval before performing the taxonomic classification with Kraken2.

6.1 Pipeline installation

```
$ conda install -c bioconda nextflow
$ nextflow pull maxibor/kraken-nf
```

6.2 Running the pipeline

See the README of maxibor/kraken-nf

CHAPTER 7

Adding new sources to an existing source file

```
[1]: import pandas as pd
     import numpy as np
```

```
[24]: def add_new_data(old_data, new_data, old_labels, label):
          """
          Update the sourcepredict learning table
          INPUT:
              old_data(str): path to csv file of existing sourcepredict source data table
              new_data(str): path to csv file of new TAXID count table, with TAXID as 1st␣
      ↪column
              old_labels(str): path to sourcepredict csv file of labels
              label(str): scientific name of new sample's specie. Example: 'Sus_scrofa'
          OUTPUT:
              merged(pd.DataFrame): merged old and new source data table for sourcepredict
              labels(pd.DataFrame): updated labels data table
          """
          old = pd.read_csv(old_data, index_col=0)
          old = old.drop(['labels'], axis = 0)
          new = pd.read_csv(new_data)
          merged = pd.merge(left=old, right=new, how='outer', on='TAXID')
          merged = merged.fillna(0)
          old_labels = pd.read_csv(old_labels, index_col=0)
          new_labels = pd.DataFrame([label]*(new.shape[1]-1), new.columns[1:])
          new_labels.columns=['labels']
          labels = old_labels.append(new_labels)
          return(merged, labels)
```

```
[30]: labs = add_new_data(old_data=old_data, new_data=new_data, old_labels=old_labels,␣
      ↪label=label)[1]
```

```
[31]: labs.to_csv("new_sources.csv")
```

CHAPTER 8

Sourcepredict example 1: Gut host species prediction

```
[1]: import pandas as pd
     import pandas_ml as pdml
```

In this example, we will use Sourcepredict and Sourcetracker2 applied to the example dataset provided in the Sourcepredict directory.

The example datasets contains the following samples: - *Homo sapiens* gut microbiome (1, 2, 3, 4, 5, 6) - *Canis familiaris* gut microbiome (1) - Soil microbiome (1, 2, 3)

8.1 Preparing the data

```
[2]: cnt = pd.read_csv("../data/modern_gut_microbiomes_sources.csv", index_col=0)
     labels = pd.read_csv("../data/modern_gut_microbiomes_labels.csv", index_col=0)
```

This is a TAXID count table containing the samples as columns headers, and the TAXID as row indices

```
[3]: cnt.head()
```

```
[3]:        SRR1175007   SRR042182   SRR061154   SRR061499  SRR063469   SRR062324  \
     TAXID
     0        3528337.0  11563613.0  10084261.0  20054993.0  8747525.0  12116517.0
     6              0.0        78.0         0.0       127.0        0.0        79.0
     7              0.0        78.0         0.0       127.0        0.0        79.0
     9              0.0       129.0         0.0       153.0        0.0       151.0
     10             0.0       160.0         0.0       193.0        0.0        99.0

            SRR1179037   SRR061236   SRR061456   SRR642021  ...  mgm4477903_3  \
     TAXID                                                  ...
     0        4191329.0  13992760.0  14825759.0  11083673.0  ...     6169203.0
     6              0.0         0.0         0.0       172.0  ...          68.0
     7              0.0         0.0         0.0       172.0  ...          68.0
     9              0.0       165.0        96.0         0.0  ...           0.0
```

(continues on next page)

(continued from previous page)

```
10            0.0        55.0       249.0       238.0  ...           0.0

       mgm4477807_3  mgm4477874_3  mgm4477904_3  mgm4477804_3  mgm4477873_3  \
TAXID
0         8820851.0     5713837.0    10238500.0     5055930.0    10380594.0
6             247.0         211.0         156.0         147.0         383.0
7             247.0         211.0         156.0         147.0         383.0
9               0.0           0.0           0.0           0.0           0.0
10              0.0           0.0           0.0           0.0           0.0

       ERR1939166  SRR3578625  ERR1939165  SRR3578645
TAXID
0      13391896.0      1553.0  14802198.0       736.0
6          1353.0         0.0      1522.0         0.0
7          1353.0         0.0      1522.0         0.0
9            77.0         0.0        65.0         0.0
10          263.0         0.0       466.0         0.0

[5 rows x 432 columns]
```

The labels file contains the mapping of samples names with their actual origin (sources)

[4]:
```
labels.head()
```

[4]:
```
                  labels
SRR1175007  Homo_sapiens
SRR042182   Homo_sapiens
SRR061154   Homo_sapiens
SRR061499   Homo_sapiens
SRR063469   Homo_sapiens
```

We will divide the source in training (95%) and testing (5%) dataset

[5]:
```
cnt_train = cnt.sample(frac=0.95, axis=1)
cnt_test = cnt.drop(cnt_train.columns, axis=1)
```

We also have to subset the labels file to only the training dataset

[6]:
```
train_labels = labels.loc[cnt_train.columns,:]
test_labels = labels.loc[cnt_test.columns,:]
```

8.2 Sourcepredict

Last but not least, we must export the files to `csv` to run sourcepredict

[7]:
```
cnt_train.to_csv("gut_species_sources.csv")
cnt_test.to_csv("gut_species_sinks.csv")
train_labels.to_csv("gut_species_labels.csv")
```

We'll now launch sourcepredict with the GMPR normalization method, and the t-SNE embedding, on 6 cores.

[8]:
```
%%time
!sourcepredict -s gut_species_sources.csv \
               -l gut_species_labels.csv \
```

(continues on next page)

(continued from previous page)

```
                -n GMPR \
                -m TSNE \
                -e example_embedding.csv \
                -t 6 gut_species_sinks.csv
```

```
Step 1: Checking for unknown proportion
  == Sample: SRR1175007 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR1175007
                 known:98.68%
                 unknown:1.32%
  == Sample: SRR061236 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR061236
                 known:85.01%
                 unknown:14.99%
  == Sample: SRR063471 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR063471
                 known:98.4%
                 unknown:1.6%
  == Sample: SRR1930132 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR1930132
                 known:98.48%
                 unknown:1.52%
  == Sample: SRR1930133 ==
        Adding unknown
        Normalizing (GMPR)
```

(continues on next page)

(continued from previous page)

```
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.99
        ----------------------
        - Sample: SRR1930133
                 known:98.49%
                 unknown:1.51%
== Sample: SRR7658586 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR7658586
                 known:79.65%
                 unknown:20.35%
== Sample: SRR7658645 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.99
        ----------------------
        - Sample: SRR7658645
                 known:26.88%
                 unknown:73.12%
== Sample: SRR7658584 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.98
        ----------------------
        - Sample: SRR7658584
                 known:85.78%
                 unknown:14.22%
== Sample: SRR7658607 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.99
        ----------------------
        - Sample: SRR7658607
                 known:98.74%
                 unknown:1.26%
```

(continues on next page)

(continued from previous page)

```
== Sample: SRR7658597 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 0.98
      ----------------------
      - Sample: SRR7658597
               known:99.1%
               unknown:0.9%
== Sample: SRR5898944 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5898944
               known:98.48%
               unknown:1.52%
== Sample: ERR1914439 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: ERR1914439
               known:98.48%
               unknown:1.52%
== Sample: ERR1915140 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: ERR1915140
               known:98.48%
               unknown:1.52%
== Sample: ERR1914041 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
```

(continues on next page)

(continued from previous page)

```
      - Sample: ERR1914041
               known:98.48%
               unknown:1.52%
== Sample: ERR1915022 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: ERR1915022
               known:98.48%
               unknown:1.52%
== Sample: ERR1915826 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: ERR1915826
               known:98.48%
               unknown:1.52%
== Sample: ERR1913400 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: ERR1913400
               known:98.48%
               unknown:1.52%
== Sample: ERR1915765 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: ERR1915765
               known:98.48%
               unknown:1.52%
== Sample: ERR1915225 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
```

(continues on next page)

(continued from previous page)

```
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: ERR1915225
                 known:98.48%
                 unknown:1.52%
  == Sample: mgm4477874_3 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mgm4477874_3
                 known:72.37%
                 unknown:27.63%
  == Sample: ERR1939166 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.97
        ----------------------
        - Sample: ERR1939166
                 known:47.44%
                 unknown:52.56%
  == Sample: ERR1939165 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.97
        ----------------------
        - Sample: ERR1939165
                 known:55.27%
                 unknown:44.73%
Step 2: Checking for source proportion
        Computing weighted_unifrac distance on species rank
        TSNE embedding in 2 dimensions
        KNN machine learning
        Performing 5 fold cross validation on 6 cores...
        Trained KNN classifier with 10 neighbors
        -> Testing Accuracy: 0.99
        ----------------------
        - Sample: SRR1175007
                 Canis_familiaris:1.81%
                 Homo_sapiens:96.72%
                 Soil:1.47%
        - Sample: SRR061236
                 Canis_familiaris:1.81%
                 Homo_sapiens:96.72%
```

(continues on next page)

(continued from previous page)

```
                Soil:1.47%
        - Sample: SRR063471
                Canis_familiaris:1.81%
                Homo_sapiens:96.72%
                Soil:1.47%
        - Sample: SRR1930132
                Canis_familiaris:1.81%
                Homo_sapiens:96.72%
                Soil:1.47%
        - Sample: SRR1930133
                Canis_familiaris:1.81%
                Homo_sapiens:96.72%
                Soil:1.47%
        - Sample: SRR7658586
                Canis_familiaris:1.81%
                Homo_sapiens:96.72%
                Soil:1.47%
        - Sample: SRR7658645
                Canis_familiaris:1.81%
                Homo_sapiens:96.72%
                Soil:1.47%
        - Sample: SRR7658584
                Canis_familiaris:1.81%
                Homo_sapiens:96.72%
                Soil:1.47%
        - Sample: SRR7658607
                Canis_familiaris:1.81%
                Homo_sapiens:96.72%
                Soil:1.47%
        - Sample: SRR7658597
                Canis_familiaris:1.81%
                Homo_sapiens:96.72%
                Soil:1.47%
        - Sample: SRR5898944
                Canis_familiaris:1.81%
                Homo_sapiens:96.72%
                Soil:1.47%
        - Sample: ERR1914439
                Canis_familiaris:94.25%
                Homo_sapiens:4.28%
                Soil:1.47%
        - Sample: ERR1915140
                Canis_familiaris:94.25%
                Homo_sapiens:4.28%
                Soil:1.47%
        - Sample: ERR1914041
                Canis_familiaris:94.25%
                Homo_sapiens:4.28%
                Soil:1.47%
        - Sample: ERR1915022
                Canis_familiaris:94.25%
                Homo_sapiens:4.28%
                Soil:1.47%
        - Sample: ERR1915826
                Canis_familiaris:94.25%
                Homo_sapiens:4.28%
                Soil:1.47%
```

(continues on next page)

(continued from previous page)

```
        - Sample: ERR1913400
                Canis_familiaris:94.25%
                Homo_sapiens:4.28%
                Soil:1.47%
        - Sample: ERR1915765
                Canis_familiaris:94.25%
                Homo_sapiens:4.28%
                Soil:1.47%
        - Sample: ERR1915225
                Canis_familiaris:94.25%
                Homo_sapiens:4.28%
                Soil:1.47%
        - Sample: mgm4477874_3
                Canis_familiaris:2.51%
                Homo_sapiens:5.95%
                Soil:91.53%
        - Sample: ERR1939166
                Canis_familiaris:2.51%
                Homo_sapiens:5.95%
                Soil:91.53%
        - Sample: ERR1939165
                Canis_familiaris:2.51%
                Homo_sapiens:5.95%
                Soil:91.53%
Sourcepredict result written to gut_species_sinks.sourcepredict.csv
Embedding coordinates written to example_embedding.csv
CPU times: user 3.64 s, sys: 828 ms, total: 4.47 s
Wall time: 4min 2s
```

Two files were generated by Sourcepredict: - `gut_species_sinks.sourcepredict.csv` which contains the proportions of each source

```
[9]: sourcepred = pd.read_csv("gut_species_sinks.sourcepredict.csv", index_col=0)
```

```
[10]: sourcepred
```

```
[10]:
                  SRR1175007  SRR061236  SRR063471  SRR1930132  SRR1930133  \
Canis_familiaris    0.017814   0.015347   0.017765    0.017779    0.017781
Homo_sapiens        0.954443   0.822254   0.951788    0.952581    0.952644
Soil                0.014516   0.012506   0.014476    0.014488    0.014489
unknown             0.013226   0.149893   0.015971    0.015152    0.015086

                  SRR7658586  SRR7658645  SRR7658584  SRR7658607  SRR7658597  \
Canis_familiaris    0.014379    0.004853    0.015486    0.017825    0.017891
Homo_sapiens        0.770401    0.260037    0.829699    0.955022    0.958548
Soil                0.011717    0.003955    0.012619    0.014525    0.014579
unknown             0.203503    0.731155    0.142196    0.012627    0.008983

                  ...  ERR1915140  ERR1914041  ERR1915022  ERR1915826  \
Canis_familiaris  ...    0.928216    0.928216    0.928216    0.928216
Homo_sapiens      ...    0.042128    0.042128    0.042128    0.042128
Soil              ...    0.014504    0.014504    0.014504    0.014504
unknown           ...    0.015152    0.015152    0.015152    0.015152

                  ERR1913400  ERR1915765  ERR1915225  mgm4477874_3  \
Canis_familiaris    0.928216    0.928216    0.928216      0.018200
Homo_sapiens        0.042128    0.042128    0.042128      0.043078
```

(continues on next page)

(continued from previous page)

```
Soil                   0.014504    0.014504    0.014504      0.662431
unknown                0.015152    0.015152    0.015152      0.276292

                     ERR1939166  ERR1939165
Canis_familiaris       0.011931    0.013898
Homo_sapiens           0.028241    0.032896
Soil                   0.434275    0.505859
unknown                0.525554    0.447347

[4 rows x 22 columns]
```

Let's check which organism was predicted for each samples, and compare it with the true source

[11]:
```
comparison = sourcepred.idxmax().to_frame(name="prediction").merge(test_labels, left_
↪index=True, right_index=True)
cm = pdml.ConfusionMatrix(y_true=comparison['labels'],y_pred=comparison['prediction'])
```

```
/Users/borry/miniconda3/lib/python3.6/site-packages/pandas/core/indexing.py:1494:␣
↪FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
https://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-loc-reindex-
↪listlike
  return self._getitem_tuple(key)
```

Let's look at the confusion matrix

[30]:
```
cm.to_dataframe()
```

[30]:
```
Predicted         Canis_familiaris  Homo_sapiens  Soil  unknown
Actual
Canis_familiaris                 8             0     0        0
Homo_sapiens                     0            10     0        1
Soil                             0             0     2        1
unknown                          0             0     0        0
```

Finally, let's compute the accuracy

[31]:
```
round(cm.stats()['overall']['Accuracy'],2)
```

[31]:
```
0.91
```

91% of the sink samples were correctly predicted !

- The second file generated by sourcepredict is `example_embedding.csv` which contains the embedding coordinates of all samples (sources and sinks)

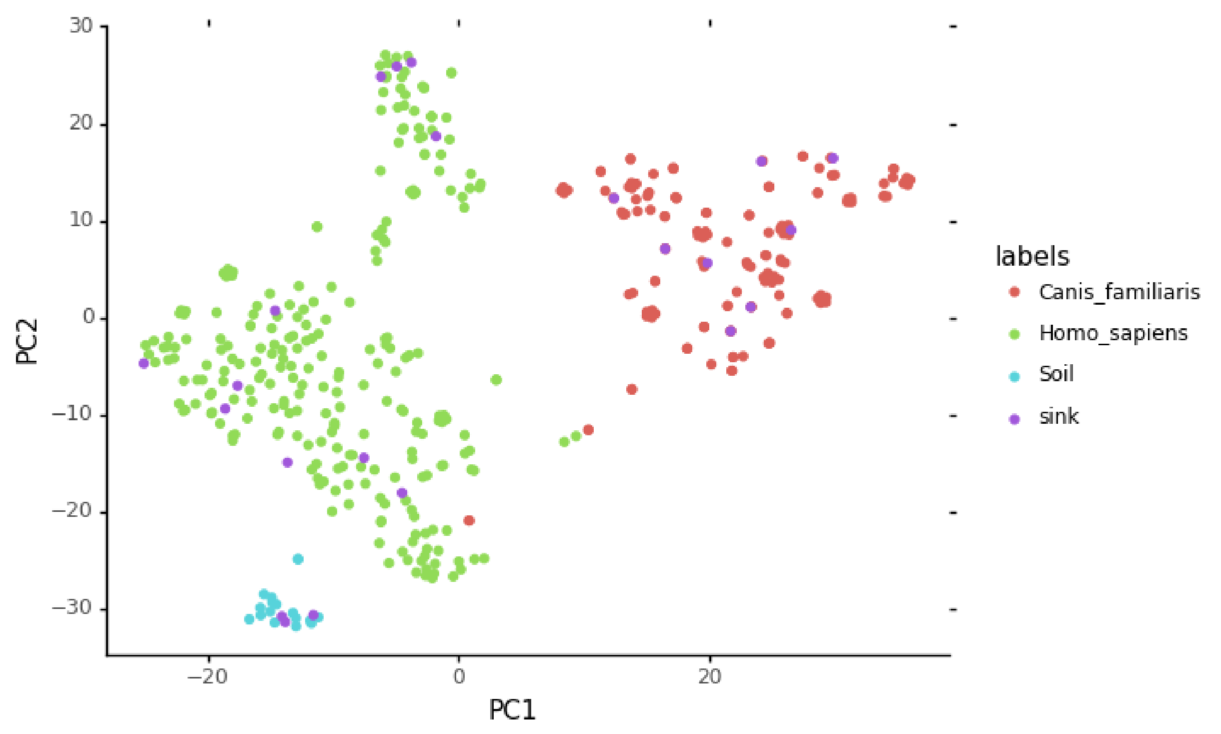
[14]:
```
embed = pd.read_csv("example_embedding.csv", index_col=0)
embed.head()
```

[14]:
```
                  PC1        PC2            labels        name
SRR1761672 -12.706208  -7.860738      Homo_sapiens  SRR1761672
SRR061456   -4.520492  24.795073      Homo_sapiens   SRR061456
SRR1761718 -20.427488  -6.425568      Homo_sapiens  SRR1761718
SRR7658589 -23.176891  -2.985772      Homo_sapiens  SRR7658589
ERR1914932  28.669333  12.863045  Canis_familiaris  ERR1914932
```

We can plot this embedding, using for example, plotnine, which implements the grammar of graphics in Python

```
[15]: from plotnine import *
      import warnings
      warnings.filterwarnings('ignore')
```

```
[16]: ggplot(data = embed, mapping = aes(x="PC1",y="PC2", color="labels")) + geom_point() +␣
      ↪theme_classic()
```



```
[16]: <ggplot: (-9223372029842878797)>
```

We can see on this plot where the sink samples were embedded

8.3 Sourcetracker2

"SourceTracker is designed to predict the source of microbial communities in a set of input samples" and is generally consired as the gold standard method to do so. The version 2 is a rewrite of the original Sourcetracker in Python.

We'll reuse the same training and test files, but we need to reformat them a bit for sourcetracker: - In sourcetracker, the source (training) and sink (file) TAXIDs count table is a single file - The metadata file is slightly different

```
[17]: cnt.to_csv("gut_species_taxid.csv", sep="\t", index_label="TAXID")
```

```
[18]: test_labels['SourceSink'] = ['sink']*test_labels.shape[0]
```

```
[19]: train_labels['SourceSink'] = ['source']*train_labels.shape[0]
```

```
[20]: metadata = train_labels.append(test_labels).rename(columns = {"labels":"Env"})[[
      ↪'SourceSink','Env']]
      metadata.head()
```

```
[20]:            SourceSink               Env
      SRR1761672     source      Homo_sapiens
      SRR061456      source      Homo_sapiens
      SRR1761718     source      Homo_sapiens
      SRR7658589     source      Homo_sapiens
      ERR1914932     source  Canis_familiaris
```

```
[21]: metadata.to_csv("st_gut_species_metadata.csv", sep="\t", index_label='#SampleID')
```

Finally, we need to convert the TAXIDs count table to biom format

```
[22]: !biom convert -i gut_species_taxid.csv -o gut_species_taxid.biom --table-type="Taxon␣
      ↪table" --to-json
```

Soucetracker launch command: `sourcetracker2 gibbs -i gut_species_taxid.biom -m st_gut_species_metadata.csv -o gut_species --jobs 6`
(Sourcetracker2 was run on a Linux remote server because of issues running it on MacOS)

```
[32]: st_pred = pd.read_csv("gut_species/mixing_proportions.txt", sep = "\t", index_col=0)
      st_pred.head()
```

```
[32]:             Canis_familiaris  Homo_sapiens    Soil  Unknown
      #SampleID
      SRR1175007            0.0170        0.9609  0.0063   0.0158
      SRR061236             0.0358        0.9365  0.0074   0.0203
      SRR063471             0.0121        0.9724  0.0032   0.0123
      SRR1930132            0.1466        0.3761  0.4477   0.0296
      SRR1930133            0.1182        0.5082  0.3507   0.0229
```

```
[34]: st_comparison = st_pred.idxmax(axis=1).to_frame(name="prediction")
      st_comparison.head()
```

```
[34]:               prediction
      #SampleID
      SRR1175007  Homo_sapiens
      SRR061236   Homo_sapiens
      SRR063471   Homo_sapiens
      SRR1930132          Soil
      SRR1930133  Homo_sapiens
```

Let’s compare the SourceTracker prediction with the true source

```
[35]: comparison2 = st_comparison.merge(test_labels, left_index=True, right_index=True)
      comparison2
```

```
[35]:               prediction        labels SourceSink
      SRR1175007  Homo_sapiens  Homo_sapiens       sink
      SRR061236   Homo_sapiens  Homo_sapiens       sink
      SRR063471   Homo_sapiens  Homo_sapiens       sink
      SRR1930132          Soil  Homo_sapiens       sink
      SRR1930133  Homo_sapiens  Homo_sapiens       sink
```

(continues on next page)

(continued from previous page)

```
SRR7658586        Homo_sapiens      Homo_sapiens       sink
SRR7658645        Homo_sapiens      Homo_sapiens       sink
SRR7658584                Soil      Homo_sapiens       sink
SRR7658607        Homo_sapiens      Homo_sapiens       sink
SRR7658597        Homo_sapiens      Homo_sapiens       sink
SRR5898944        Homo_sapiens      Homo_sapiens       sink
ERR1914439    Canis_familiaris  Canis_familiaris       sink
ERR1915140                Soil  Canis_familiaris       sink
ERR1914041    Canis_familiaris  Canis_familiaris       sink
ERR1915022    Canis_familiaris  Canis_familiaris       sink
ERR1915826    Canis_familiaris  Canis_familiaris       sink
ERR1913400    Canis_familiaris  Canis_familiaris       sink
ERR1915765    Canis_familiaris  Canis_familiaris       sink
ERR1915225    Canis_familiaris  Canis_familiaris       sink
mgm4477874_3              Soil              Soil       sink
ERR1939166                Soil              Soil       sink
ERR1939165                Soil              Soil       sink
```

Computing the accuracy

```
[36]: cm2 = pdml.ConfusionMatrix(y_true=comparison2["labels"], y_pred=comparison2[
      ↪"prediction"])
      cm2.to_dataframe()
```

```
[36]: Predicted          Canis_familiaris  Homo_sapiens  Soil
      Actual
      Canis_familiaris                  7             0     1
      Homo_sapiens                      0             9     2
      Soil                              0             0     3
```

```
[38]: acc2 = round(cm2.stats()['overall']['Accuracy'],2)
```

```
[38]: 0.86
```

Here, Sourcetracker only managed to predict 86% of the sink samples origin correctly

8.4 Conclusion

On this dataset, we've seen that Sourcepredict performs similary or even better than Sourcetracker on predicting accurately the source species

CHAPTER 9

Sourcepredict example2: Estimating source proportions

For this example, we'll reuse the dog, human, and soil dataset.

But unlike example1, here we will mix samples from different sources and estimate the mixing proportions with Sourcepredict and Sourcetracker2

9.1 Preparing mixed samples

```
[1]: import pandas as pd
     from plotnine import *
     import numpy as np
```

```
[2]: cnt = pd.read_csv("../data/modern_gut_microbiomes_sources.csv", index_col=0)
     labels = pd.read_csv("../data/modern_gut_microbiomes_labels.csv",index_col=0)
```

As in example 1, we'll first split the dataset into training (95%) and testing(5%)

```
[3]: cnt_train = cnt.sample(frac=0.95, axis=1, random_state=42)
     cnt_test = cnt.drop(cnt_train.columns, axis=1)
     train_labels = labels.loc[cnt_train.columns,:]
     test_labels = labels.loc[cnt_test.columns,:]
```

```
[4]: test_labels['labels'].value_counts()
```

```
[4]: Homo_sapiens        13
     Canis_familiaris     8
     Soil                 1
     Name: labels, dtype: int64
```

```
[5]: cnt_test.head()
```

```
[5]:        SRR059440  SRR1930140  SRR1761708  SRR1761664  SRR1761667  SRR1761674  \
     TAXID
```

(continues on next page)

(continued from previous page)

```
0       19805534.0   7267728.0  18530434.0   2460493.0   3324349.0   2835521.0
6              0.0         0.0        85.0         0.0         0.0         0.0
7              0.0         0.0        85.0         0.0         0.0         0.0
9            239.0        88.0       115.0        55.0       189.0        91.0
10           163.0       177.0       112.0        76.0       164.0       109.0

       SRR7658684  SRR7658622  SRR7658689  SRR7658619  ...  SRR5898940  \
TAXID                                                  ...
0       5565044.0  18783402.0   6319253.0  18641694.0  ...   1658949.0
6             0.0       542.0         0.0       217.0  ...         0.0
7             0.0       542.0         0.0       217.0  ...         0.0
9            72.0       100.0       152.0       209.0  ...         0.0
10          220.0        84.0       180.0       175.0  ...         0.0

       ERR1914224  ERR1915611  ERR1915293  ERR1914207  ERR1915420  ERR1916218  \
TAXID
0       3442424.0   1529589.0   1765224.0   1815426.0   1364019.0   1558043.0
6             0.0         0.0         0.0         0.0         0.0         0.0
7             0.0         0.0         0.0         0.0         0.0         0.0
9            51.0        69.0        60.0        56.0       106.0        51.0
10          202.0        80.0        67.0        73.0         0.0        80.0

       ERR1913675  ERR1914667  SRR3578645
TAXID
0       1617964.0   1557538.0       736.0
6             0.0         0.0         0.0
7             0.0         0.0         0.0
9            69.0        62.0         0.0
10           77.0       110.0         0.0

[5 rows x 22 columns]
```

We then create a function to randomly select a sample from each source (dog as s_{dog} and human as s_{human}), and combine such as the new sample $s_{mixed} = p1 * s_{dog} + p1 * s_{human}$

```
[6]: def create_mixed_sample(cnt, labels, p1, samp_name, seed):
         rand_dog = labels.query('labels == "Canis_familiaris"').sample(1, random_state =␣
     ↪seed).index[0]
         rand_human = labels.query('labels == "Homo_sapiens"').sample(1, random_state =␣
     ↪seed).index[0]
         dog_samp = cnt[rand_dog]*p1
         human_samp = cnt[rand_human]*(1-p1)
         comb = dog_samp + human_samp
         comb = comb.rename(samp_name)
         meta = pd.DataFrame({'human_sample':[rand_human],'dog_sample':[rand_dog], 'human_
     ↪prop':[(1-p1)], 'dog_prop':[p1]}, index=[samp_name])
         return(comb, meta)
```

We run this function for a range of mixed proportions (0 to 100%, by 10%), 3 time for each mix

```
[7]: mixed_samp = []
     mixed_meta = []
     nb = 1
     for i in range(3):
         for p1 in np.arange(0,1.1,0.1):
             s = create_mixed_sample(cnt=cnt_test, labels=test_labels, p1=p1, samp_name=f
     ↪"mixed_sample_{nb}", seed = int(100*p1))
```

(continues on next page)

(continued from previous page)

```
        mixed_samp.append(s[0])
        mixed_meta.append(s[1])
        nb += 1
```

[8]:
```
mixed_samples = pd.concat(mixed_samp, axis=1, keys=[s.name for s in mixed_samp]).
↪astype(int)
mixed_samples.head()
```

[8]:
```
       mixed_sample_1  mixed_sample_2  mixed_sample_3  mixed_sample_4  \
TAXID
0             5565044         2390966        15338330        14408501
6                   0               0             433               0
7                   0               0             433               0
9                  72              55              90             184
10                220              75              83             136

       mixed_sample_5  mixed_sample_6  mixed_sample_7  mixed_sample_8  \
TAXID
0             7983985         5100662         8740624         3251030
6                  66              26               0               0
7                  66              26               0               0
9                  98             141             159              74
10                276             117              65             109

       mixed_sample_9  mixed_sample_10  ...  mixed_sample_24  mixed_sample_25  \
TAXID                                   ...
0             3418809          2103402  ...          2390966         15338330
6                   0                0  ...                0              433
7                   0                0  ...                0              433
9                  78               70  ...               55               90
10                194               89  ...               75               83

       mixed_sample_26  mixed_sample_27  mixed_sample_28  mixed_sample_29  \
TAXID
0             14408501          7983985          5100662          8740624
6                    0               66               26                0
7                    0               66               26                0
9                  184               98              141              159
10                 136              276              117               65

       mixed_sample_30  mixed_sample_31  mixed_sample_32  mixed_sample_33
TAXID
0              3251030          3418809          2103402          1529589
6                    0                0                0                0
7                    0                0                0                0
9                   74               78               70               69
10                 109              194               89               80

[5 rows x 33 columns]
```

[9]:
```
mixed_metadata = pd.concat(mixed_meta)
mixed_metadata
```

[9]:
```
                human_sample  dog_sample  human_prop  dog_prop
mixed_sample_1    SRR7658684  ERR1913675         1.0       0.0
mixed_sample_2    SRR1761664  ERR1915293         0.9       0.1
mixed_sample_3    SRR7658622  ERR1916218         0.8       0.2
```

(continues on next page)

(continued from previous page)

```
mixed_sample_4      SRR059440  ERR1914207          0.7        0.3
mixed_sample_5     SRR7658624  ERR1914667          0.6        0.4
mixed_sample_6     SRR7658608  ERR1915420          0.5        0.5
mixed_sample_7      SRR059440  ERR1915420          0.4        0.6
mixed_sample_8     SRR1930140  ERR1915611          0.3        0.7
mixed_sample_9     SRR1761667  ERR1914224          0.2        0.8
mixed_sample_10    SRR1930140  ERR1915611          0.1        0.9
mixed_sample_11    SRR7658624  ERR1915611          0.0        1.0
mixed_sample_12    SRR7658684  ERR1913675          1.0        0.0
mixed_sample_13    SRR1761664  ERR1915293          0.9        0.1
mixed_sample_14    SRR7658622  ERR1916218          0.8        0.2
mixed_sample_15     SRR059440  ERR1914207          0.7        0.3
mixed_sample_16    SRR7658624  ERR1914667          0.6        0.4
mixed_sample_17    SRR7658608  ERR1915420          0.5        0.5
mixed_sample_18     SRR059440  ERR1915420          0.4        0.6
mixed_sample_19    SRR1930140  ERR1915611          0.3        0.7
mixed_sample_20    SRR1761667  ERR1914224          0.2        0.8
mixed_sample_21    SRR1930140  ERR1915611          0.1        0.9
mixed_sample_22    SRR7658624  ERR1915611          0.0        1.0
mixed_sample_23    SRR7658684  ERR1913675          1.0        0.0
mixed_sample_24    SRR1761664  ERR1915293          0.9        0.1
mixed_sample_25    SRR7658622  ERR1916218          0.8        0.2
mixed_sample_26     SRR059440  ERR1914207          0.7        0.3
mixed_sample_27    SRR7658624  ERR1914667          0.6        0.4
mixed_sample_28    SRR7658608  ERR1915420          0.5        0.5
mixed_sample_29     SRR059440  ERR1915420          0.4        0.6
mixed_sample_30    SRR1930140  ERR1915611          0.3        0.7
mixed_sample_31    SRR1761667  ERR1914224          0.2        0.8
mixed_sample_32    SRR1930140  ERR1915611          0.1        0.9
mixed_sample_33    SRR7658624  ERR1915611          0.0        1.0
```

Now we can export the new "test" (sink) table to `csv` for sourcepredict

```
[10]: mixed_samples.to_csv('mixed_samples_cnt.csv')
```

As well as the source count and labels table for the sources

```
[11]: train_labels.to_csv('train_labels.csv')
      cnt_train.to_csv('sources_cnt.csv')
```

9.2 Sourcepredict

9.2.1 With KNN machine learning

For running Sourcepredict, we'll change two parameters from their default values: - `-me` The default method used by Sourcepredict is T-SNE which a non-linear type of embedding, i.e. the distance between points doesn't reflext their actual distance in the original dimensions, to achieve a better clustering, which is good for source prediction. Because here we're more interested in source proportion estimation, rather than source prediction, we'll choose a Multi Dimensional Scaling (MDS) which is a type of linear embedding, where the distance between points in the lower dimension match more the distances in the embedding in lower dimension, which is better for source proportion estimation. - `-kne` which is the number of neighbors in KNN algorithm: we use all neighbors to reflect a more global contribution of samples to the proportion estimation, instead of only the immediate neighbors. This will affect negatively the source prediction, but give better source proportion estimations

[12]:
```
%%time
!python ../sourcepredict -s sources_cnt.csv \
               -l train_labels.csv \
               -kne all\
               -me mds \
               -e mixed_embedding.csv \
               -t 6 \
               mixed_samples_cnt.csv
```

```
/Users/borry/miniconda3/envs/sourcepredict/lib/python3.7/site-packages/sklearn/
↪externals/joblib/__init__.py:15: DeprecationWarning: sklearn.externals.joblib is␣
↪deprecated in 0.21 and will be removed in 0.23. Please import this functionality␣
↪directly from joblib, which can be installed with: pip install joblib. If this␣
↪warning is raised when loading pickled models, you may need to re-serialize those␣
↪models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
Step 1: Checking for unknown proportion
  == Sample: mixed_sample_1 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_1
                 known:98.48%
                 unknown:1.52%
  == Sample: mixed_sample_2 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_2
                 known:98.48%
                 unknown:1.52%
  == Sample: mixed_sample_3 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.98
        ----------------------
        - Sample: mixed_sample_3
                 known:98.49%
                 unknown:1.51%
  == Sample: mixed_sample_4 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
```

(continues on next page)

(continued from previous page)

```
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_4
                 known:98.49%
                 unknown:1.51%
== Sample: mixed_sample_5 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_5
                 known:98.48%
                 unknown:1.52%
== Sample: mixed_sample_6 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_6
                 known:98.48%
                 unknown:1.52%
== Sample: mixed_sample_7 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_7
                 known:98.48%
                 unknown:1.52%
== Sample: mixed_sample_8 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_8
                 known:98.48%
                 unknown:1.52%
== Sample: mixed_sample_9 ==
        Adding unknown
```

(continues on next page)

(continued from previous page)

```
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_9
                known:98.48%
                unknown:1.52%
== Sample: mixed_sample_10 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_10
                known:98.48%
                unknown:1.52%
== Sample: mixed_sample_11 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_11
                known:98.48%
                unknown:1.52%
== Sample: mixed_sample_12 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_12
                known:98.48%
                unknown:1.52%
== Sample: mixed_sample_13 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_13
                known:98.48%
```

(continues on next page)

(continued from previous page)

```
                unknown:1.52%
== Sample: mixed_sample_14 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.98
        ----------------------
        - Sample: mixed_sample_14
                known:98.49%
                unknown:1.51%
== Sample: mixed_sample_15 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_15
                known:98.49%
                unknown:1.51%
== Sample: mixed_sample_16 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_16
                known:98.48%
                unknown:1.52%
== Sample: mixed_sample_17 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_17
                known:98.48%
                unknown:1.52%
== Sample: mixed_sample_18 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
```

(continues on next page)

(continued from previous page)

```
      ----------------------
      - Sample: mixed_sample_18
              known:98.48%
              unknown:1.52%
== Sample: mixed_sample_19 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: mixed_sample_19
              known:98.48%
              unknown:1.52%
== Sample: mixed_sample_20 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: mixed_sample_20
              known:98.48%
              unknown:1.52%
== Sample: mixed_sample_21 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: mixed_sample_21
              known:98.48%
              unknown:1.52%
== Sample: mixed_sample_22 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: mixed_sample_22
              known:98.48%
              unknown:1.52%
== Sample: mixed_sample_23 ==
      Adding unknown
      Normalizing (GMPR)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
```

(continues on next page)

(continued from previous page)

```
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_23
                 known:98.48%
                 unknown:1.52%
== Sample: mixed_sample_24 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_24
                 known:98.48%
                 unknown:1.52%
== Sample: mixed_sample_25 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.98
        ----------------------
        - Sample: mixed_sample_25
                 known:98.49%
                 unknown:1.51%
== Sample: mixed_sample_26 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_26
                 known:98.49%
                 unknown:1.51%
== Sample: mixed_sample_27 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_27
                 known:98.48%
                 unknown:1.52%
== Sample: mixed_sample_28 ==
        Adding unknown
```

(continues on next page)

(continued from previous page)

```
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_28
                 known:98.48%
                 unknown:1.52%
  == Sample: mixed_sample_29 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_29
                 known:98.48%
                 unknown:1.52%
  == Sample: mixed_sample_30 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_30
                 known:98.48%
                 unknown:1.52%
  == Sample: mixed_sample_31 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_31
                 known:98.48%
                 unknown:1.52%
  == Sample: mixed_sample_32 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_32
                 known:98.48%
```

(continues on next page)

(continued from previous page)

```
                 unknown:1.52%
  == Sample: mixed_sample_33 ==
        Adding unknown
        Normalizing (GMPR)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: mixed_sample_33
                 known:98.48%
                 unknown:1.52%
Step 2: Checking for source proportion
        Computing weighted_unifrac distance on species rank
        MDS embedding in 2 dimensions
        KNN machine learning
        Trained KNN classifier with 262 neighbors
        -> Testing Accuracy: 0.91
        ----------------------
        - Sample: mixed_sample_1
                 Canis_familiaris:26.67%
                 Homo_sapiens:72.03%
                 Soil:1.3%
        - Sample: mixed_sample_2
                 Canis_familiaris:22.23%
                 Homo_sapiens:76.47%
                 Soil:1.3%
        - Sample: mixed_sample_3
                 Canis_familiaris:19.26%
                 Homo_sapiens:78.12%
                 Soil:2.62%
        - Sample: mixed_sample_4
                 Canis_familiaris:25.91%
                 Homo_sapiens:72.03%
                 Soil:2.06%
        - Sample: mixed_sample_5
                 Canis_familiaris:23.01%
                 Homo_sapiens:73.37%
                 Soil:3.62%
        - Sample: mixed_sample_6
                 Canis_familiaris:22.79%
                 Homo_sapiens:75.82%
                 Soil:1.39%
        - Sample: mixed_sample_7
                 Canis_familiaris:25.65%
                 Homo_sapiens:72.46%
                 Soil:1.89%
        - Sample: mixed_sample_8
                 Canis_familiaris:48.84%
                 Homo_sapiens:49.91%
                 Soil:1.24%
        - Sample: mixed_sample_9
                 Canis_familiaris:33.12%
                 Homo_sapiens:65.14%
                 Soil:1.75%
        - Sample: mixed_sample_10
```

(continues on next page)

(continued from previous page)

```
                 Canis_familiaris:62.75%
                 Homo_sapiens:36.18%
                 Soil:1.06%
        - Sample: mixed_sample_11
                 Canis_familiaris:80.93%
                 Homo_sapiens:18.45%
                 Soil:0.61%
        - Sample: mixed_sample_12
                 Canis_familiaris:26.67%
                 Homo_sapiens:72.03%
                 Soil:1.3%
        - Sample: mixed_sample_13
                 Canis_familiaris:22.23%
                 Homo_sapiens:76.47%
                 Soil:1.3%
        - Sample: mixed_sample_14
                 Canis_familiaris:19.26%
                 Homo_sapiens:78.12%
                 Soil:2.62%
        - Sample: mixed_sample_15
                 Canis_familiaris:25.91%
                 Homo_sapiens:72.03%
                 Soil:2.06%
        - Sample: mixed_sample_16
                 Canis_familiaris:23.01%
                 Homo_sapiens:73.37%
                 Soil:3.62%
        - Sample: mixed_sample_17
                 Canis_familiaris:22.79%
                 Homo_sapiens:75.82%
                 Soil:1.39%
        - Sample: mixed_sample_18
                 Canis_familiaris:25.65%
                 Homo_sapiens:72.46%
                 Soil:1.89%
        - Sample: mixed_sample_19
                 Canis_familiaris:48.84%
                 Homo_sapiens:49.91%
                 Soil:1.24%
        - Sample: mixed_sample_20
                 Canis_familiaris:33.12%
                 Homo_sapiens:65.14%
                 Soil:1.75%
        - Sample: mixed_sample_21
                 Canis_familiaris:62.75%
                 Homo_sapiens:36.18%
                 Soil:1.06%
        - Sample: mixed_sample_22
                 Canis_familiaris:80.93%
                 Homo_sapiens:18.45%
                 Soil:0.61%
        - Sample: mixed_sample_23
                 Canis_familiaris:26.67%
                 Homo_sapiens:72.03%
                 Soil:1.3%
        - Sample: mixed_sample_24
                 Canis_familiaris:22.23%
```

(continues on next page)

(continued from previous page)

```
                 Homo_sapiens:76.47%
                 Soil:1.3%
        - Sample: mixed_sample_25
                 Canis_familiaris:19.26%
                 Homo_sapiens:78.12%
                 Soil:2.62%
        - Sample: mixed_sample_26
                 Canis_familiaris:25.91%
                 Homo_sapiens:72.03%
                 Soil:2.06%
        - Sample: mixed_sample_27
                 Canis_familiaris:23.01%
                 Homo_sapiens:73.37%
                 Soil:3.62%
        - Sample: mixed_sample_28
                 Canis_familiaris:22.79%
                 Homo_sapiens:75.82%
                 Soil:1.39%
        - Sample: mixed_sample_29
                 Canis_familiaris:25.65%
                 Homo_sapiens:72.46%
                 Soil:1.89%
        - Sample: mixed_sample_30
                 Canis_familiaris:48.84%
                 Homo_sapiens:49.91%
                 Soil:1.24%
        - Sample: mixed_sample_31
                 Canis_familiaris:33.12%
                 Homo_sapiens:65.14%
                 Soil:1.75%
        - Sample: mixed_sample_32
                 Canis_familiaris:62.75%
                 Homo_sapiens:36.18%
                 Soil:1.06%
        - Sample: mixed_sample_33
                 Canis_familiaris:80.93%
                 Homo_sapiens:18.45%
                 Soil:0.61%
Sourcepredict result written to mixed_samples_cnt.sourcepredict.csv
Embedding coordinates written to mixed_embedding.csv
CPU times: user 4.27 s, sys: 1.14 s, total: 5.41 s
Wall time: 5min 46s
```

Reading Sourcepredict KNN results

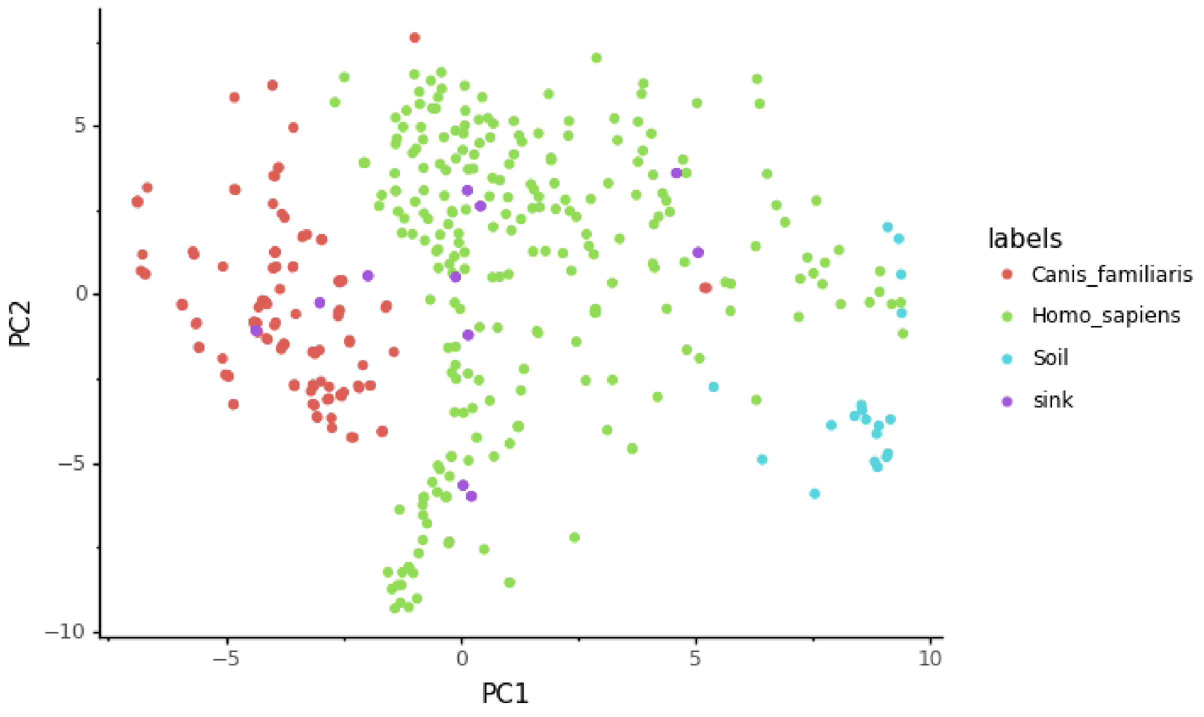
[13]:
```
sp_ebd = pd.read_csv("mixed_embedding.csv", index_col=0)
```

[14]:
```
sp_ebd.head()
```

[14]:
```
                  PC1       PC2            labels          name
mgm4477874_3  8.822395 -4.957090              Soil  mgm4477874_3
SRR1761709    3.896029  6.258361      Homo_sapiens    SRR1761709
SRR7658685   -1.151347  5.457706      Homo_sapiens    SRR7658685
SRR059395    -0.889409 -7.682652      Homo_sapiens     SRR059395
ERR1915122   -3.533856 -2.673234  Canis_familiaris    ERR1915122
```

```
[15]: import warnings
      warnings.filterwarnings('ignore')
```

```
[16]: ggplot(data = sp_ebd, mapping = aes(x='PC1',y='PC2')) + geom_point(aes(color='labels
      ↪')) + theme_classic()
```



```
[16]: <ggplot: (297174606)>
```

```
[17]: sp_pred = pd.read_csv("mixed_samples_cnt.sourcepredict.csv", index_col=0)
```

```
[18]: sp_pred.T.head()
```

```
[18]:                 Canis_familiaris  Homo_sapiens      Soil   unknown
      mixed_sample_1          0.262665      0.709352  0.012832  0.015152
      mixed_sample_2          0.218900      0.753113  0.012836  0.015152
      mixed_sample_3          0.189678      0.769471  0.025776  0.015075
      mixed_sample_4          0.255186      0.709368  0.020298  0.015148
      mixed_sample_5          0.226581      0.722568  0.035699  0.015152
```

```
[19]: mixed_metadata.head()
```

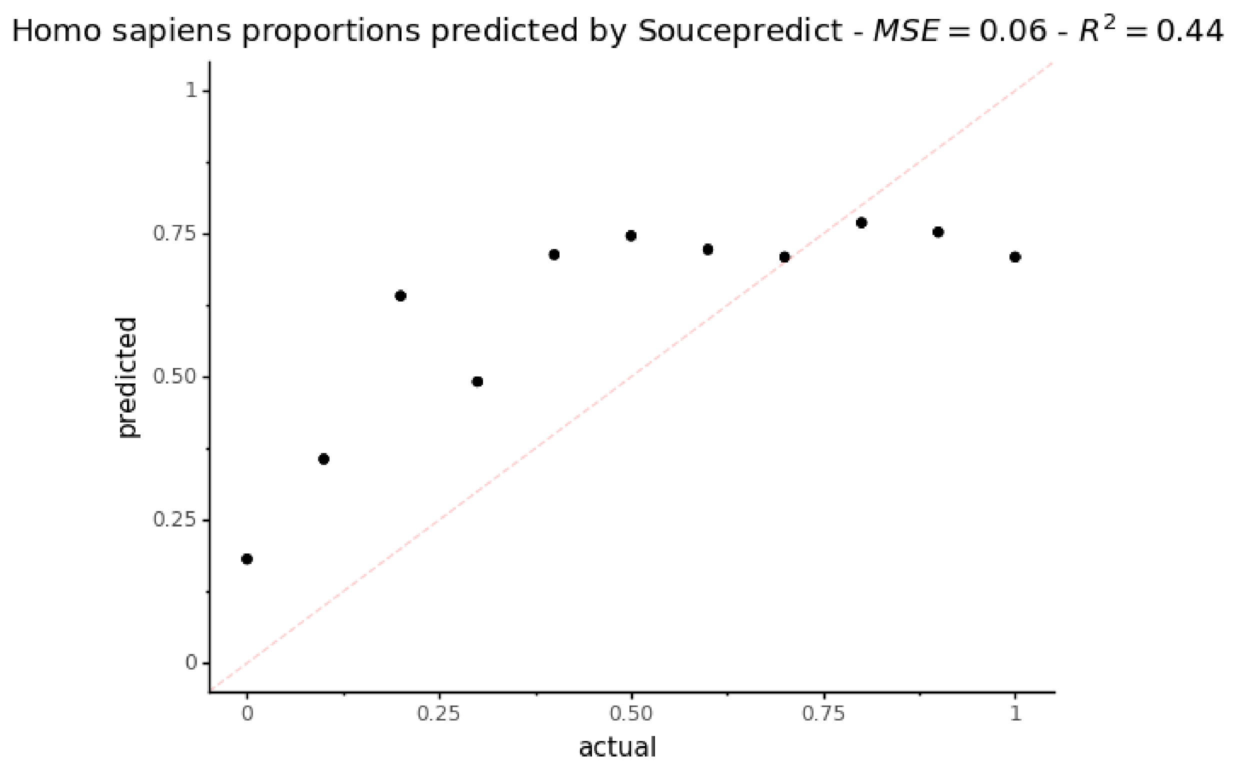
```
[19]:                human_sample  dog_sample  human_prop  dog_prop
      mixed_sample_1   SRR7658684  ERR1913675         1.0       0.0
      mixed_sample_2   SRR1761664  ERR1915293         0.9       0.1
      mixed_sample_3   SRR7658622  ERR1916218         0.8       0.2
      mixed_sample_4    SRR059440  ERR1914207         0.7       0.3
      mixed_sample_5   SRR7658624  ERR1914667         0.6       0.4
```

```
[20]: sp_res = sp_pred.T.merge(mixed_metadata, left_index=True, right_index=True)
```

```
[21]: from sklearn.metrics import r2_score, mean_squared_error
```

```
[22]: mse_sp = round(mean_squared_error(y_pred=sp_res['Homo_sapiens'], y_true=sp_res['human_
      ↪prop']),2)
      r2_sp = round(r2_score(y_pred=sp_res['Homo_sapiens'], y_true=sp_res['human_prop']),2)
```

```
[23]: p = ggplot(data = sp_res, mapping=aes(x='human_prop',y='Homo_sapiens')) + geom_point()
      p += labs(title = f"Homo sapiens proportions predicted by Soucepredict - $MSE = {mse_
      ↪sp}$ - $R^2 = {r2_sp}$", x='actual', y='predicted')
      p += theme_classic()
      p += coord_cartesian(xlim=[0,1], ylim=[0,1])
      p += geom_abline(intercept=0, slope=1, color = "red", alpha=0.2, linetype = 'dashed')
      p
```

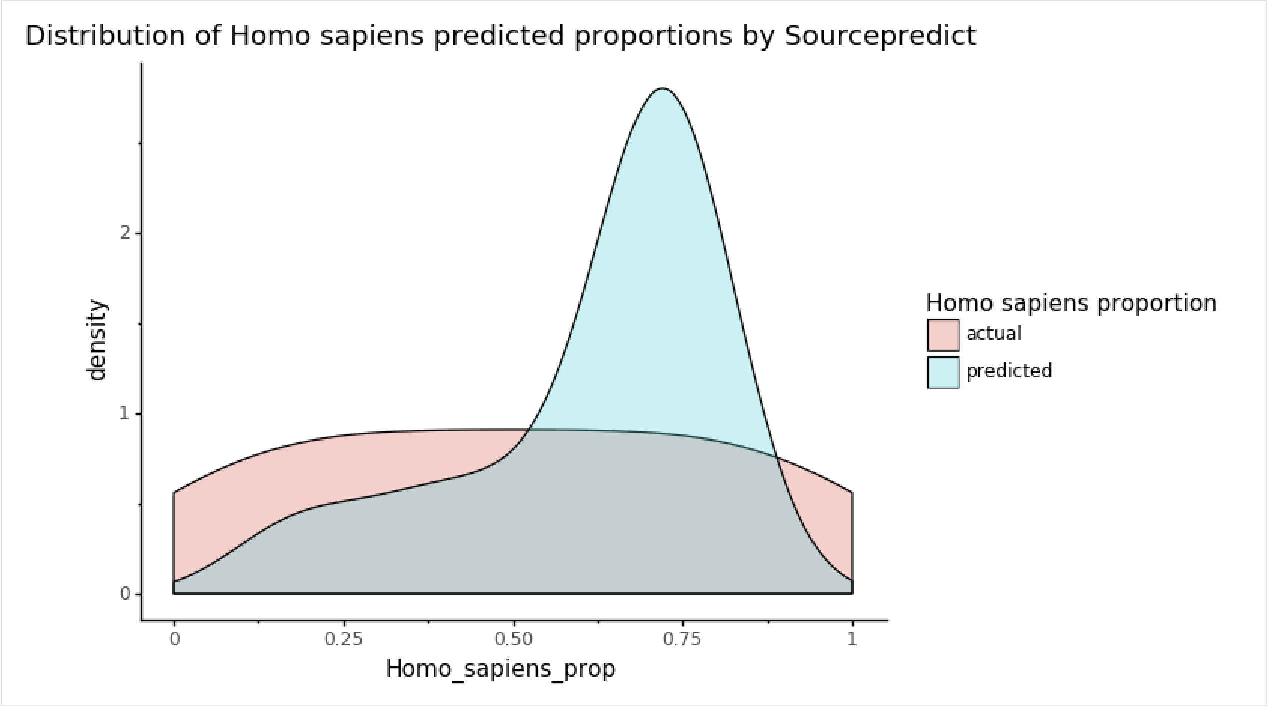


```
[23]: <ggplot: (-9223372036557649955)>
```

On this plot, the dotted red line represents what a perfect proportion estimation would give

```
[24]: sp_res_hist = (sp_res['human_prop'].append(sp_res['Homo_sapiens']).to_frame(name=
      ↪'Homo_sapiens_prop'))
      sp_res_hist['source'] = (['actual']*sp_res.shape[0]+['predicted']*sp_res.shape[0])
```

```
[25]: p = ggplot(data = sp_res_hist, mapping=aes(x='Homo_sapiens_prop')) + geom_
      ↪density(aes(fill='source'), alpha=0.3)
      p += labs(title = 'Distribution of Homo sapiens predicted proportions by Sourcepredict
      ↪')
      p += scale_fill_discrete(name="Homo sapiens proportion")
      p += theme_classic()
      p
```



```
[25]: <ggplot: (-9223372036557649934)>
```

This plot shows the actual and predicted by Sourcepredict distribution of Human proportions. What we are interested in is the overlap between the two colors: the higher it is, the more the estimated Human proportion is accurate.

9.3 Sourcetracker2

Preparing count table

```
[26]: cnt_train.merge(mixed_samples, right_index=True, left_index=True).to_csv("st_mixed_
      ↪count.csv" , sep="\t", index_label="TAXID")
```

```
[27]: !biom convert -i st_mixed_count.csv -o st_mixed_count.biom --table-type="Taxon table"␣
      ↪--to-json
```

Preparing metadata

```
[28]: train_labels['SourceSink'] = ['source']*train_labels.shape[0]
```

```
[29]: mixed_metadata['labels'] = ['-']*mixed_metadata.shape[0]
      mixed_metadata['SourceSink'] = ['sink']*mixed_metadata.shape[0]
```

```
[30]: st_labels = train_labels.append(mixed_metadata[['labels', 'SourceSink']])
```

```
[31]: st_labels = st_labels.rename(columns={'labels':'Env'})[['SourceSink','Env']]
```

```
[32]: st_labels.to_csv("st_mixed_labels.csv", sep="\t", index_label='#SampleID')
```

Running Sourcetracker2 `sourcetracker2 gibbs -i st_mixed_count.biom -m st_mixed_labels.csv -o mixed_prop --jobs 6`
(Sourcetracker2 was run on a Linux remote server because of issues running it on MacOS)

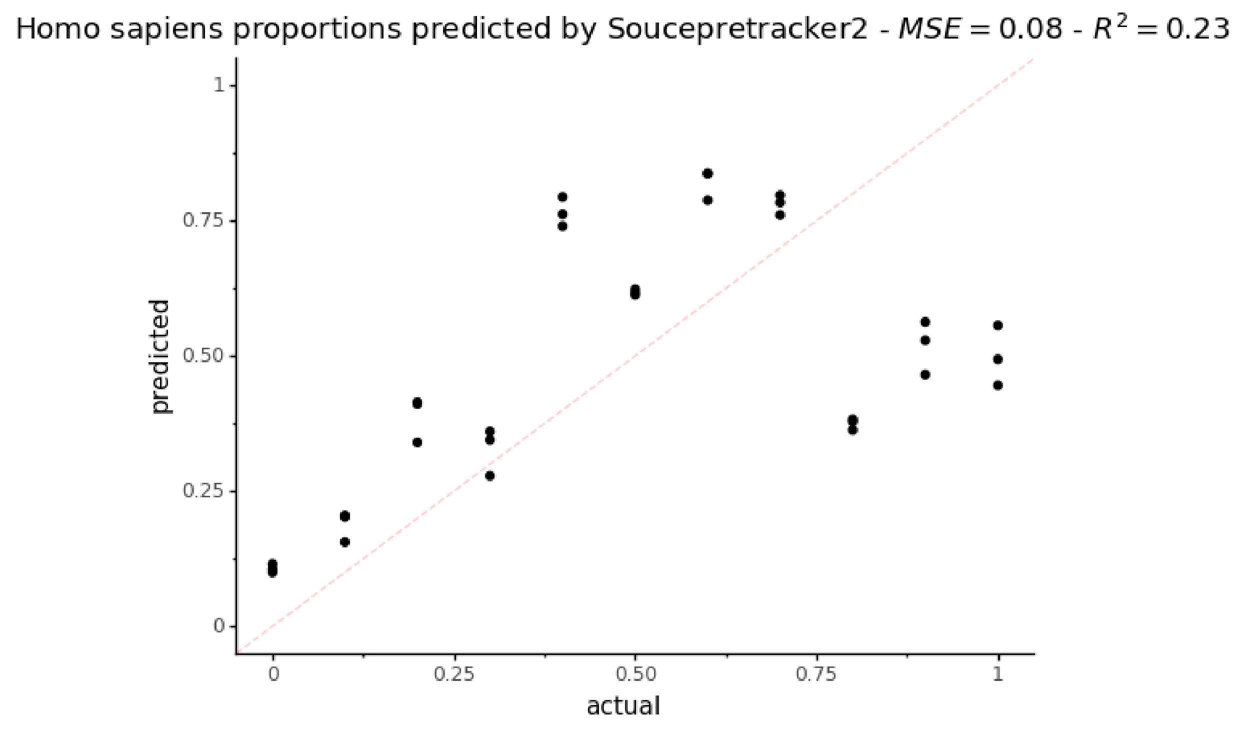
Sourcetracker2 results

```
[33]: st_pred = pd.read_csv("_assets/mixed_prop/mixing_proportions.txt", sep="\t", index_
      ↪col=0)
```

```
[34]: st_res = st_pred.merge(mixed_metadata, left_index=True, right_index=True)
```

```
[35]: mse_st = round(mean_squared_error(y_pred=st_res['Homo_sapiens'], y_true=st_res['human_
      ↪prop']),2)
      r2_st = round(r2_score(y_pred=st_res['Homo_sapiens'], y_true=st_res['human_prop']),2)
```

```
[36]: p = ggplot(data = st_res, mapping=aes(x='human_prop',y='Homo_sapiens')) + geom_point()
      p += labs(title = f"Homo sapiens proportions predicted by Soucepretracker2 - $MSE =
      ↪{mse_st}$ - $R^2 = {r2_st}$", x='actual', y='predicted')
      p += theme_classic()
      p += coord_cartesian(xlim=[0,1], ylim=[0,1])
      p += geom_abline(intercept=0, slope=1, color = "red", alpha=0.2, linetype = 'dashed')
      p
```

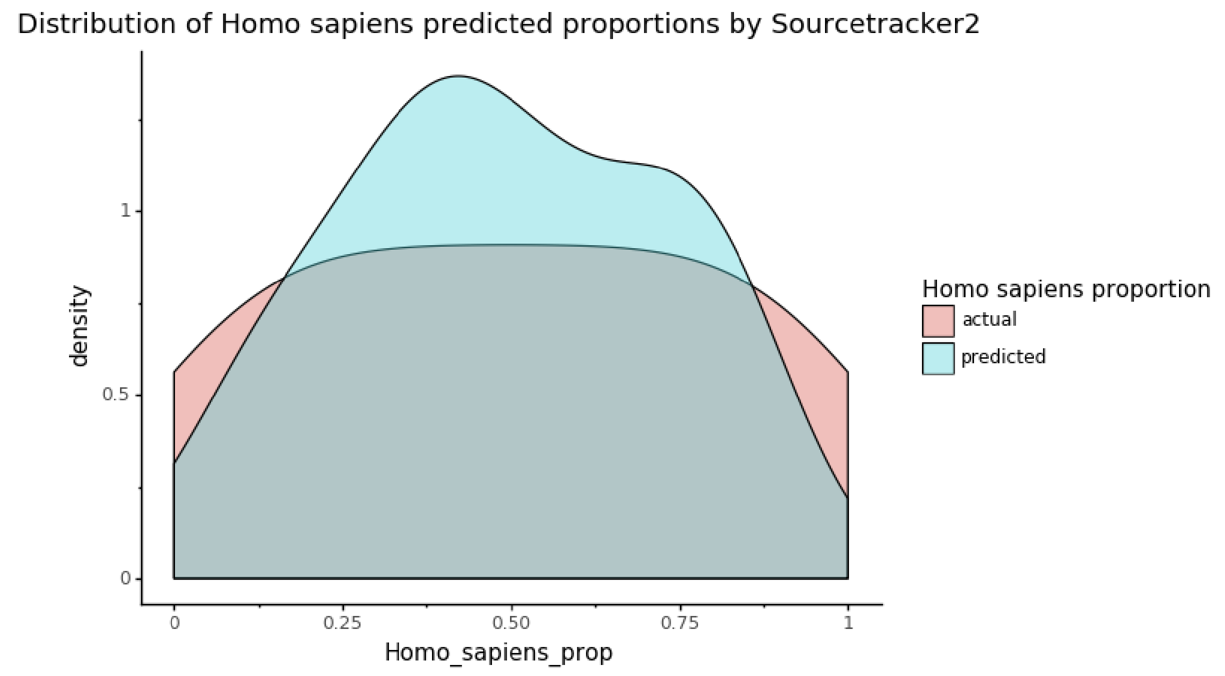


```
[36]: <ggplot: (297644629)>
```

On this plot, the dotted red line represents what a perfect proportion estimation would give.

```
[37]: st_res_hist = (st_res['human_prop'].append(st_res['Homo_sapiens']).to_frame(name=
      ↪'Homo_sapiens_prop'))
      st_res_hist['source'] = (['actual']*st_res.shape[0]+['predicted']*st_res.shape[0])
```

```
[38]: p = ggplot(data = st_res_hist, mapping=aes(x='Homo_sapiens_prop')) + geom_
      ↪density(aes(fill='source'), alpha=0.4)
      p += labs(title = 'Distribution of Homo sapiens predicted proportions by␣
      ↪Sourcetracker2')
      p += scale_fill_discrete(name="Homo sapiens proportion")
      p += theme_classic()
      p
```



```
[38]: <ggplot: (297182405)>
```

This plot shows the actual and predicted by Sourcepredict distribution of Human proportions. What we are interested in is the overlap between the two colors: the higher it is, the more the estimated Human proportion is accurate.

9.4 Conclusion

For source proportion estimation in samples of mixed sources, Sourcepredict, especially when using it with `-kne all` neighbors, performs similarly, or slightly better than Sourcetracker2.

However, Sourcepredict wasn't designed for source prediction in mind, as opposed to source proportion estimation. Therefore, for source proportion estimation, we still recommend using Sourcetracker2, even if Sourcepredict can perform similarly.

CHAPTER 10

Sourcepredict example3: Segregating patients with or without *Clostridium difficile* infection (CDI) on the basis of 16s microbiome

Source Article: Domestic canines do not display evidence of gut microbial dysbiosis in the presence of Clostridioides (Clostridium) difficile, despite cellular susceptibility to its toxins 10.1016/j.anaerobe.2019.03.017

Healthy human dataset: PRJNA386260
CDI human dataset: PRJNA307992

```
[1]: import pandas as pd
     import numpy as np
     from plotnine import *
     from ete3 import NCBITaxa
     import multiprocessing
     from functools import partial
     import seaborn as sns
```

```
[2]: ncbi = NCBITaxa()
```

10.1 Downloading data

```
[3]: cdi_color = "#E7CE1A"
     healthy_color = "grey"
```

```
[4]: tax_level = ['genus','species']
```

```
[5]: import multiprocessing
     import subprocess
```

(continues on next page)

(continued from previous page)

```
def dl(file, outdir):
    cmd = f"wget {file} -P {outdir}"
    print(cmd)
    try:
        subprocess.check_output(cmd, shell=True)
    except subprocess.CalledProcessError:
        print(f"Error downloading {file}")


def dl_multi(allfiles, outdir, process):
    dl_fun = partial(dl, outdir=outdir)
    with multiprocessing.Pool(process) as p:
        p.map(dl_fun, allfiles)
```

```
[6]: healthy_meta = pd.read_csv("healthy/PRJNA386260_metadata.txt", sep="\t", index_col=
↪'run_accession')
CDI_meta = pd.read_csv("CDI/PRJNA307992_metadata.txt", sep="\t", index_col='run_
↪accession')
```

```
[7]: healthy_meta['labels'] = ['healthy']*healthy_meta.shape[0]
CDI_meta['labels'] = ['CDI']*CDI_meta.shape[0]
```

```
[8]: healthy_fastqs = list(healthy_meta['fastq_ftp'].str.split(";", expand=True)[0]) +␣
↪list(healthy_meta['fastq_ftp'].str.split(";", expand=True)[1])
```

```
[9]: CDI_fastqs = list(CDI_meta['fastq_ftp'].str.split(";", expand=True)[0]) + list(CDI_
↪meta['fastq_ftp'].str.split(";", expand=True)[1])
```

Uncomment to download files

```
[10]: #dl_multi(allfiles=healthy_fastqs, outdir="./healthy/", process=4)
#dl_multi(allfiles=CDI_fastqs, outdir="./CDI/", process=4)
```

10.1.1 Utility functions

Removing outlier samples (less than 10 species) and species present in less than 10 samples

```
[11]: def remove_outlier(df, n=10):
    return(df.loc[df.nunique(axis=1) > n, df.nunique(axis=0) > n])
```

Removing TAXID not in NCBI taxonomy

```
[12]: def remove_not_taxo(df):
    """
    df(pandas DataFrame) with TAXID in index, and samples in columns
    """
    valid_ranks = {k:v for (k,v) in zip(ncbi.get_rank(df.index).keys(), ncbi.get_
↪rank(df.index).values()) if v != 'no rank'}
    return(df.loc[valid_ranks.keys(),:])
```

Normalization methods

```
[13]: def gmpr_size_factor(col, ar):
          """Generate GMPR size factor
          Args:
              col (int): columm index of the numpy array
              ar (numpy array): numpy array of TAXID counts,
                  colums as Samples, Rows as TAXIDs
          Returns:
              float: GMPR size factor per column
          """
          pr = np.apply_along_axis(lambda x: np.divide(ar[:, col], x), 0, ar)
          pr[np.isinf(pr)] = np.nan
          pr[pr == 0] = np.nan
          pr_median = np.nanmedian(pr, axis=0)
          return(np.exp(np.mean(np.log(pr_median))))


      def GMPR_normalize(df, process=4):
          """Compute GMPR normalization
          Global Mean of Pairwise Ratios
          Chen, L., Reeve, J., Zhang, L., Huang, S., Wang, X., & Chen, J. (2018).
          GMPR: A robust normalization method for zero-inflated count data
          with application to microbiome sequencing data.
          PeerJ, 6, e4600.
          Args:
              df (pandas Dataframe): TAXID count dataframe,
                  colums as Samples, Rows as TAXIDs
              process (int): number of process for parallelization
          """
          ar = np.asarray(df)

          gmpr_sf_partial = partial(gmpr_size_factor, ar=ar)
          with multiprocessing.Pool(process) as p:
              sf = p.map(gmpr_sf_partial, list(range(np.shape(ar)[1])))

          return(pd.DataFrame(np.divide(ar, sf), index=df.index, columns=df.columns))
```

```
[14]: def RLE_normalize(pd_dataframe):
          """Normalize with Relative Log Expression
          Args:
              pd_dataframe (pandas DataFrame): TAXID count dataframe,
                  colums as Samples, Rows as TAXIDs
          Returns:
              pandas DataFrame: RLE Normalized datafrane. Colums as Samples, Rows as TAXIDs
          Example:
              >>> RLE_normalize(pd.DataFrame)
          """

          step1 = pd_dataframe.apply(np.log, 0)
          step2 = step1.apply(np.average, 1)
          step3 = step2[step2.replace([np.inf, -np.inf], np.nan).notnull()]
          step4_1 = step1[step1.replace(
              [np.inf, -np.inf], np.nan).notnull().all(axis=1)]
          step4 = step4_1.subtract(step3, 0)
          step5 = step4.apply(np.median, 0)
          step6 = step5.apply(np.exp)
          step7 = pd_dataframe.divide(step6, 1).apply(round, 1)
          return(step7)
```

[15]:
```
def subsample_normalize_pd(pd_dataframe):
    """Normalize with Subsampling
    Args:
        pd_dataframe (pandas DataFrame): TAXID count dataframe,
            colums as Samples, Rows as TAXIDs
    Returns:
       pandas DataFrame: Subsample Normalized dataframe. Colums as Samples, Rows as␣
↪TAXIDs
    """

    def subsample_normalize(serie, omax):
        """Subsample normalization column wise
        imin: minimum of input range
        imax: maximum of input range
        omin: minimum of output range
        omax: maximum of output range
        x in [imin, imax]
        f(x) in [omin, omax]
                 x - imin
        f(x) = ------------ x(omax - omin) + omin
               imax - imin
        Args:
            serie (pandas Series): Indivudal Sample Column
            omax (int): maximum of output range
        Returns:
            pandas Series: normalized pandas Series
        """

        imin = min(serie)
        imax = max(serie)
        omin = 0
        if imax > 0:
            newserie = serie.apply(lambda x: (
                (x - imin)/(imax - imin)*(omax-omin)+omin))
        else:
            newserie = serie
        return(newserie)

    step1 = pd_dataframe.apply(max, 1)
    themax = max(step1)

    step2 = pd_dataframe.apply(
        subsample_normalize, axis=0, args=(themax,))
    step3 = step2.apply(np.floor, axis=1)
    return(step3)
```

PLS-DA with sklearn

[16]:
```
class plsda:
    def __init__(self, X,Y, labels):
        """
        X(pd DataFrame) normalized feature matrix with samples in index, and features␣
↪in columns
        Y(np 1D array) binary response variable encoding the grouping for each sample
        labels(named pd Series) of group label for each sample
        """
        from sklearn.cross_decomposition import PLSRegression
```

(continues on next page)

(continued from previous page)

```
        self.plsr = PLSRegression(n_components=2)
        self.plsr.fit(X, Y)
        self.scores = pd.DataFrame(self.plsr.x_scores_, index=X.index, columns=['DIM1
↪','DIM2'])
        self.scores = self.scores.join(labels['labels'])
        self.weights = pd.DataFrame(self.plsr.x_weights_, index=X.columns, columns=[
↪'DIM1','DIM2']).sort_values('DIM1', ascending=False)
        self.weights['name'] = ncbi.get_taxid_translator(self.weights.index).values()
        self.top_weights = self.weights.head(20).append(self.weights.tail(20))
        self.top_weights['name'] = pd.Categorical(self.top_weights['name'],␣
↪categories=self.top_weights['name'])
```

mds with sklearn

```
[17]: class mds:
          def __init__(self, X, labels, metric='braycurtis'):
              """
              X(pd DataFrame) normalized feature matrix with samples in index, and features␣
      ↪in columns
              labels(named pd Series) of group label for each sample
              """
              from sklearn.metrics import pairwise_distances
              from sklearn.manifold import MDS
              dist = pairwise_distances(X, metric=metric)
              self.mds = MDS(n_components=2, dissimilarity='precomputed')
              self.mds.fit(X=dist)
              self.embedding = pd.DataFrame(self.mds.embedding_, columns=['DIM1','DIM2'],␣
      ↪index=X.index)
              self.embedding = self.embedding.join(labels)
```

10.2 Reading the results of the dada2-nf pipeline

10.2.1 1- Species level

```
[18]: healthy_otu_s = pd.read_csv("/projects1/users/borry/30_dada2-nf/results_healthy/
      ↪merged/dada2_otu_table.csv", index_col=0)
      CDI_otu_s = pd.read_csv("/projects1/users/borry/30_dada2-nf/results_CDI/merged/dada2_
      ↪otu_table.csv", index_col=0)
```

```
[19]: healthy_otu_s = healthy_otu_s.drop([0], axis=0)
      CDI_otu_s = CDI_otu_s.drop([0], axis=0)
```

```
[20]: all_otu_s = healthy_otu_s.merge(CDI_otu_s, left_index=True, right_index=True)
      all_otu_s = remove_outlier(all_otu_s)
      all_otu_s.shape
```

```
[20]: (103, 305)
```

```
[21]: all_otu_s.head()
```

```
[21]:      SRR5578998  SRR5579099  SRR5579045  SRR5578981  SRR5579095  SRR5579054  \
      199         0.0         0.0         0.0         0.0         0.0         0.0
      820       745.0       725.0       973.0       708.0       710.0       507.0
```

(continues on next page)

(continued from previous page)

```
821      1103.0       742.0      2126.0         0.0      1502.0      1193.0
824         0.0         0.0         0.0         0.0         0.0         0.0
851         0.0         0.0         0.0         4.0         0.0         0.0

     SRR5578909  SRR5578907  SRR5578965  SRR5579021  ...  SRR3102417  \
199         0.0         0.0         0.0         0.0  ...         0.0
820       483.0       223.0        43.0       313.0  ...         0.0
821      1316.0        52.0       462.0       947.0  ...         0.0
824         0.0         0.0         0.0         0.0  ...         0.0
851         0.0         0.0         0.0         0.0  ...       868.0

     SRR3102498  SRR3102362  SRR3102525  SRR3102486  SRR3102556  SRR3102547  \
199         0.0         0.0         0.0         0.0         0.0         0.0
820      6319.0      1663.0         0.0         0.0         0.0         0.0
821         8.0      4899.0         0.0         0.0         0.0      7318.0
824         0.0         0.0         0.0         0.0         0.0         0.0
851         0.0         0.0         0.0         0.0         0.0      2190.0

     SRR3102572  SRR3102515  SRR3102517
199         0.0         0.0         0.0
820       350.0      2235.0        31.0
821       833.0         0.0         0.0
824         0.0         0.0         0.0
851         0.0         0.0         0.0

[5 rows x 305 columns]
```

10.2.2 2- Genus Level

[22]:
```
healthy_otu_g = pd.read_csv("/projects1/users/borry/30_dada2-nf/results_healthy_genus/
↪merged/dada2_otu_table.csv", index_col=0)
cdi_otu_g = pd.read_csv("/projects1/users/borry/30_dada2-nf/results_CDI_genus/merged/
↪dada2_otu_table.csv", index_col=0)
```

[23]:
```
healthy_otu_g = healthy_otu_g.drop([0], axis=0)
cdi_otu_g = cdi_otu_g.drop([0], axis=0)
```

[24]:
```
all_otu_g = healthy_otu_g.merge(cdi_otu_g, left_index=True, right_index=True)
all_otu_g = remove_outlier(all_otu_g)
print(all_otu_g.shape)
all_otu_g.head()
```

```
(114, 405)
```

[24]:
```
     SRR5578998  SRR5579099  SRR5579045  SRR5578981  SRR5579095  SRR5579054  \
194         0.0         0.0         0.0         0.0         0.0         6.0
286         0.0         0.0         0.0         0.0         0.0         0.0
469         0.0         0.0         0.0         0.0         0.0        17.0
482         0.0         0.0         0.0         0.0         0.0         0.0
544         0.0         0.0      1203.0         0.0         0.0         0.0

     SRR5578909  SRR5578907  SRR5578965  SRR5579115  ...  SRR3102507  \
194         0.0         0.0         0.0         0.0  ...         0.0
286         0.0         0.0         0.0         0.0  ...         0.0
469         0.0         0.0         0.0         0.0  ...         0.0
```

(continues on next page)

(continued from previous page)

```
482          0.0         0.0         0.0         0.0  ...         0.0
544          0.0       182.0         0.0         0.0  ...         0.0

     SRR3102439  SRR3102474  SRR3102487  SRR3102547  SRR3102572  SRR3102381  \
194         0.0         0.0         2.0         2.0         0.0         0.0
286        17.0         0.0         0.0         0.0         0.0         0.0
469         9.0         0.0         0.0         0.0         0.0         0.0
482         0.0         0.0         0.0         0.0         0.0         4.0
544         0.0         0.0         0.0         0.0         0.0         0.0

     SRR3102515  SRR3102405  SRR3102517
194         0.0         0.0         0.0
286         0.0         0.0         0.0
469         0.0         0.0         0.0
482         0.0         0.0         0.0
544         0.0         0.0         0.0

[5 rows x 405 columns]
```

10.3 Normalizing dataframes

```
[25]: X_s = subsample_normalize_pd(all_otu_s).T
```

```
[26]: X_g = GMPR_normalize(all_otu_g, 4).dropna(axis=1).T
```

```
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪ipykernel_launcher.py:10: RuntimeWarning: divide by zero encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪ipykernel_launcher.py:10: RuntimeWarning: invalid value encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪ipykernel_launcher.py:10: RuntimeWarning: invalid value encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪ipykernel_launcher.py:10: RuntimeWarning: divide by zero encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪ipykernel_launcher.py:10: RuntimeWarning: divide by zero encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪ipykernel_launcher.py:10: RuntimeWarning: invalid value encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪ipykernel_launcher.py:10: RuntimeWarning: divide by zero encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪ipykernel_launcher.py:10: RuntimeWarning: invalid value encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪numpy/lib/nanfunctions.py:1115: RuntimeWarning: All-NaN slice encountered
  overwrite_input=overwrite_input)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪numpy/lib/nanfunctions.py:1115: RuntimeWarning: All-NaN slice encountered
```

(continues on next page)

(continued from previous page)

```
  overwrite_input=overwrite_input)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪numpy/lib/nanfunctions.py:1115: RuntimeWarning: All-NaN slice encountered
  overwrite_input=overwrite_input)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪numpy/lib/nanfunctions.py:1115: RuntimeWarning: All-NaN slice encountered
  overwrite_input=overwrite_input)
```

10.4 Creating labels dataframe and response variable array

[27]:
```
labels = healthy_meta['labels'].to_frame().append(CDI_meta['labels'].to_frame())
```

[28]:
```
labels_s = labels.loc[X_s.index, :]
Y_s = np.where(labels_s['labels']=='healthy', '1','0')
```

[29]:
```
labels_g = labels.loc[X_g.index, :]
Y_g = np.where(labels_g['labels']=='healthy', '1','0')
```

10.5 Exploring the dataset

[30]:
```
import seaborn as sns
```

PLS-DA with Python

[31]:
```
plsda_s = plsda(X_s, Y_s, labels_s)
```

[32]:
```
plsda_s.scores
```

[32]:
```
                DIM1      DIM2   labels
SRR5578998  0.935718  0.126756  healthy
SRR5579099  2.614041  0.075360  healthy
SRR5579045 -0.424088  0.029255  healthy
SRR5578981  1.068008  0.688370  healthy
SRR5579095 -0.458098  0.082241  healthy
...              ...       ...      ...
SRR3102556 -5.292148 -2.756433      CDI
SRR3102547 -1.763199  0.052028      CDI
SRR3102572 -1.139385 -1.983172      CDI
SRR3102515 -3.229330 -3.165894      CDI
SRR3102517 -2.452133 -0.887940      CDI

[305 rows x 3 columns]
```

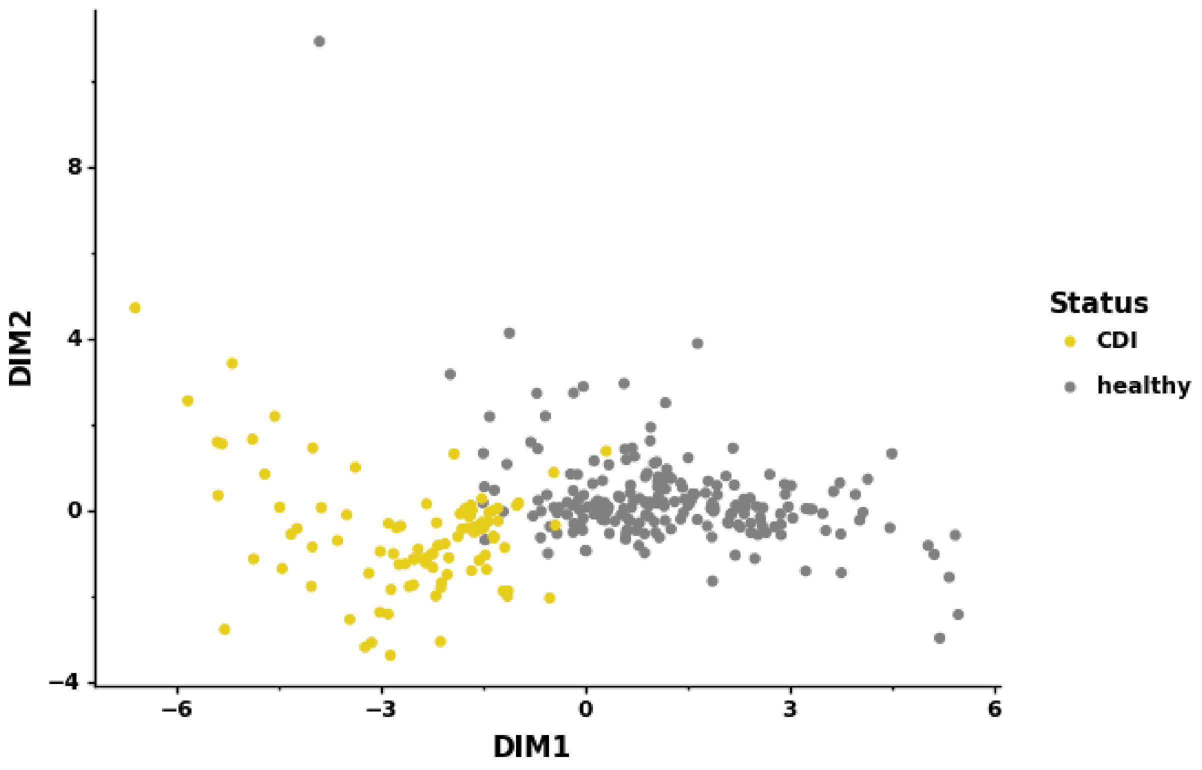
[33]:
```
g = ggplot(plsda_s.scores, aes(x='DIM1',y='DIM2', color='labels'))
g += geom_point()
g += scale_color_manual(name='Status',values = {"CDI":cdi_color, "healthy":healthy_
↪color})
g += theme_classic()
g += theme(plot_background=element_blank(),
           panel_background=element_blank(),
```

(continues on next page)

(continued from previous page)

```
           legend_background=element_blank(),
           axis_line=element_line(color='black'),
           legend_text=element_text(color='black', weight='bold'),
          axis_text=element_text(color='black', weight='bold'),
          axis_title=element_text(color='black', weight='bold'),
          legend_title=element_text(color='black', weight='bold'))
g.save('results/PLS-DA.png', format='png', dpi=300, transparent=True)
g
```

```
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:729: PlotnineWarning: Saving 6.4 x 4.8 in image.
  from_inches(height, units), units), PlotnineWarning)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:730: PlotnineWarning: Filename: results/PLS-DA.png
  warn('Filename: {}'.format(filename), PlotnineWarning)
```



```
[33]: <ggplot: (-9223363254321219135)>
```

The separation appears clearly in the first latent variable (DIM1)

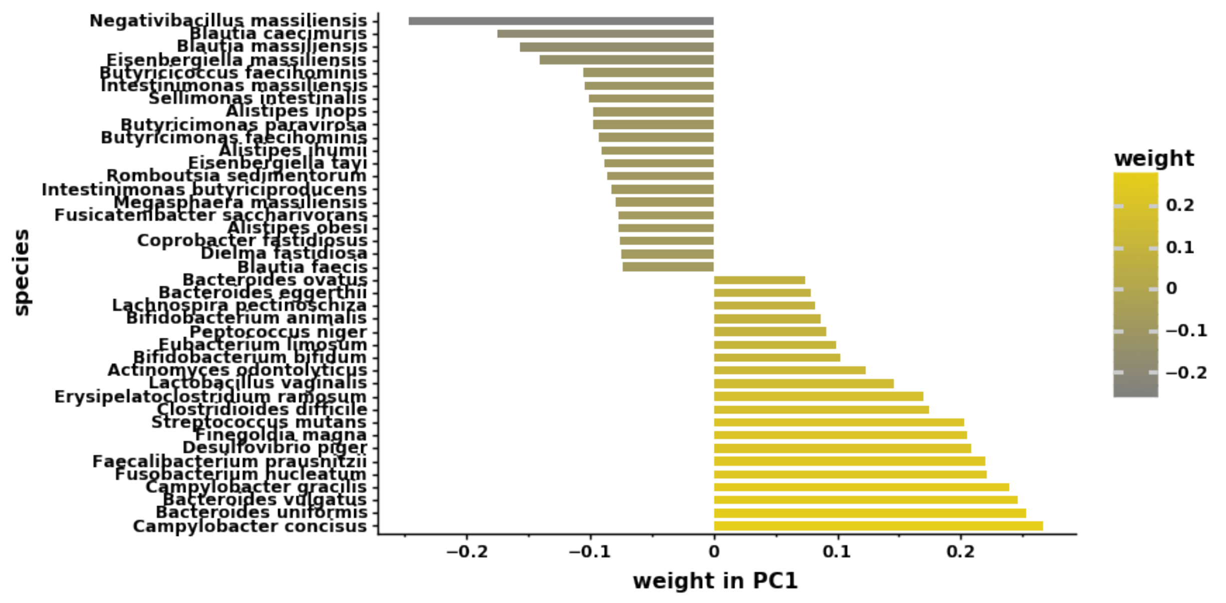
```
[34]: g = ggplot(plsda_s.top_weights, aes(x='name',y='DIM1', fill='DIM1'))
g += geom_bar(stat='identity', width=0.7)
g += coord_flip()
g += scale_fill_gradient(name = 'weight', low=healthy_color, high=cdi_color)
g += xlab('species')
g += ylab('weight in PC1')
g += theme_classic()
g += theme(plot_background=element_blank(),
           panel_background=element_blank(),
           legend_background=element_blank(),
```

(continues on next page)

(continued from previous page)

```
           axis_line=element_line(color='black'),
           legend_text=element_text(color='black', weight='bold'),
          axis_text=element_text(color='black', weight='bold'),
          axis_title=element_text(color='black', weight='bold'),
          legend_title=element_text(color='black', weight='bold'))
g.save('results/weight_CDI.png', format='png', dpi=300, transparent=True)
g
```

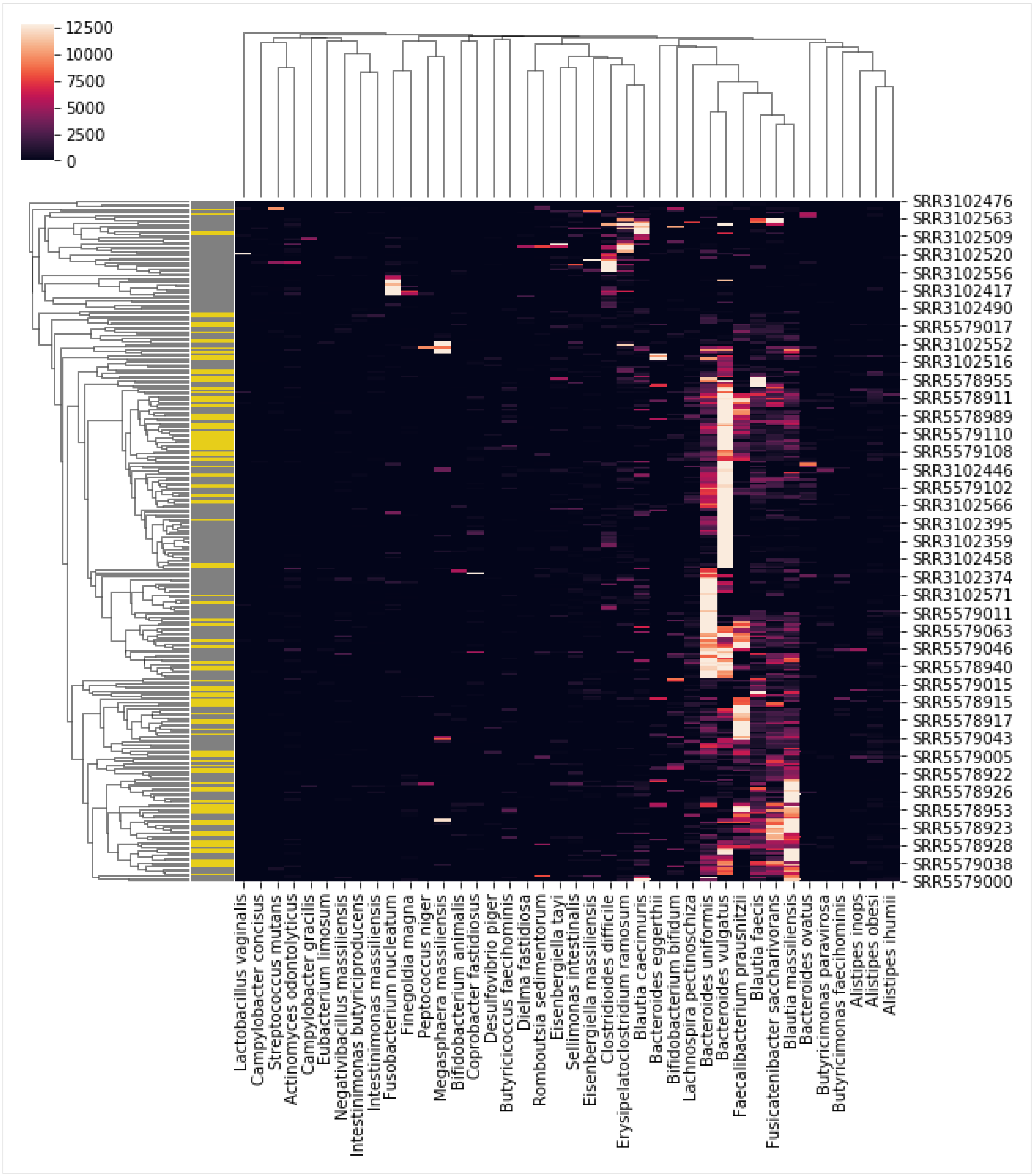
```
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:729: PlotnineWarning: Saving 6.4 x 4.8 in image.
  from_inches(height, units), units), PlotnineWarning)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:730: PlotnineWarning: Filename: results/weight_CDI.png
  warn('Filename: {}'.format(filename), PlotnineWarning)
```



```
[34]: <ggplot: (8782533497590)>
```

```
[35]: X_s_heatmap = X_s.merge(labels_s, left_index=True, right_index=True).sort_values(
      ↪'labels').drop('labels', axis=1)
      X_s_heatmap.columns = ncbi.get_taxid_translator(X_s_heatmap.columns).values()
      samp_colors = list(np.where(labels_s['labels'] == 'CDI', cdi_color,healthy_color))
      sns.clustermap(X_s_heatmap.loc[:,plsda_s.top_weights['name']], row_colors=samp_colors,
      ↪ metric='braycurtis')
```

```
[35]: <seaborn.matrix.ClusterGrid at 0x7fcd7ca32278>
```



```
[36]: labels_s
```

```
[36]:              labels
SRR5578998  healthy
SRR5579099  healthy
SRR5579045  healthy
SRR5578981  healthy
SRR5579095  healthy
```

(continues on next page)

(continued from previous page)

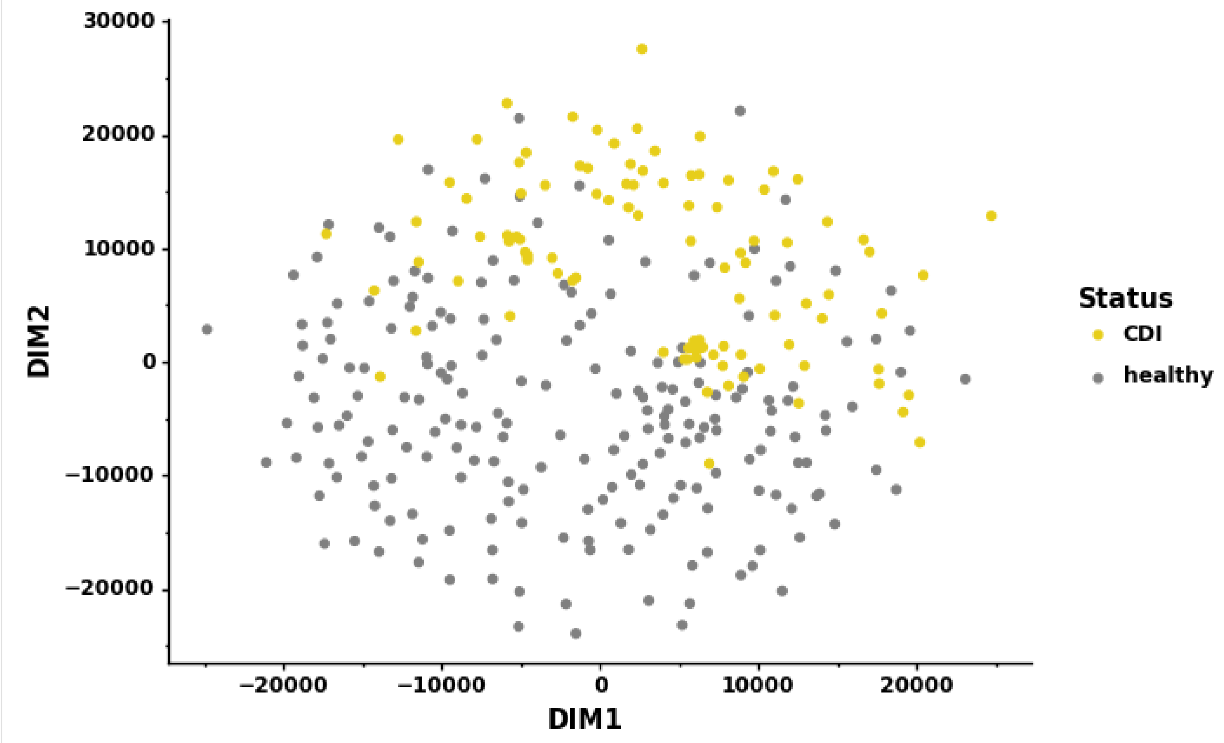
```
...             ...
SRR3102556      CDI
SRR3102547      CDI
SRR3102572      CDI
SRR3102515      CDI
SRR3102517      CDI

[305 rows x 1 columns]
```

[37]:
```
mds_s = mds(X_s, labels_s['labels'], metric='euclidean')
```

[38]:
```
g = ggplot(mds_s.embedding, aes(x='DIM1',y='DIM2', color='labels'))
g += geom_point()
g += scale_color_manual(name='Status',values = {"CDI":cdi_color, "healthy":healthy_
↪color})
g += theme_classic()
g += theme(plot_background=element_blank(),
           panel_background=element_blank(),
           legend_background=element_blank(),
           axis_line=element_line(color='black'),
           legend_text=element_text(color='black', weight='bold'),
          axis_text=element_text(color='black', weight='bold'),
          axis_title=element_text(color='black', weight='bold'),
          legend_title=element_text(color='black', weight='bold'))
g.save('results/PCoA.png', format='png', dpi=300, transparent=True)
g
```

```
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:729: PlotnineWarning: Saving 6.4 x 4.8 in image.
  from_inches(height, units), units), PlotnineWarning)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:730: PlotnineWarning: Filename: results/PCoA.png
  warn('Filename: {}'.format(filename), PlotnineWarning)
```

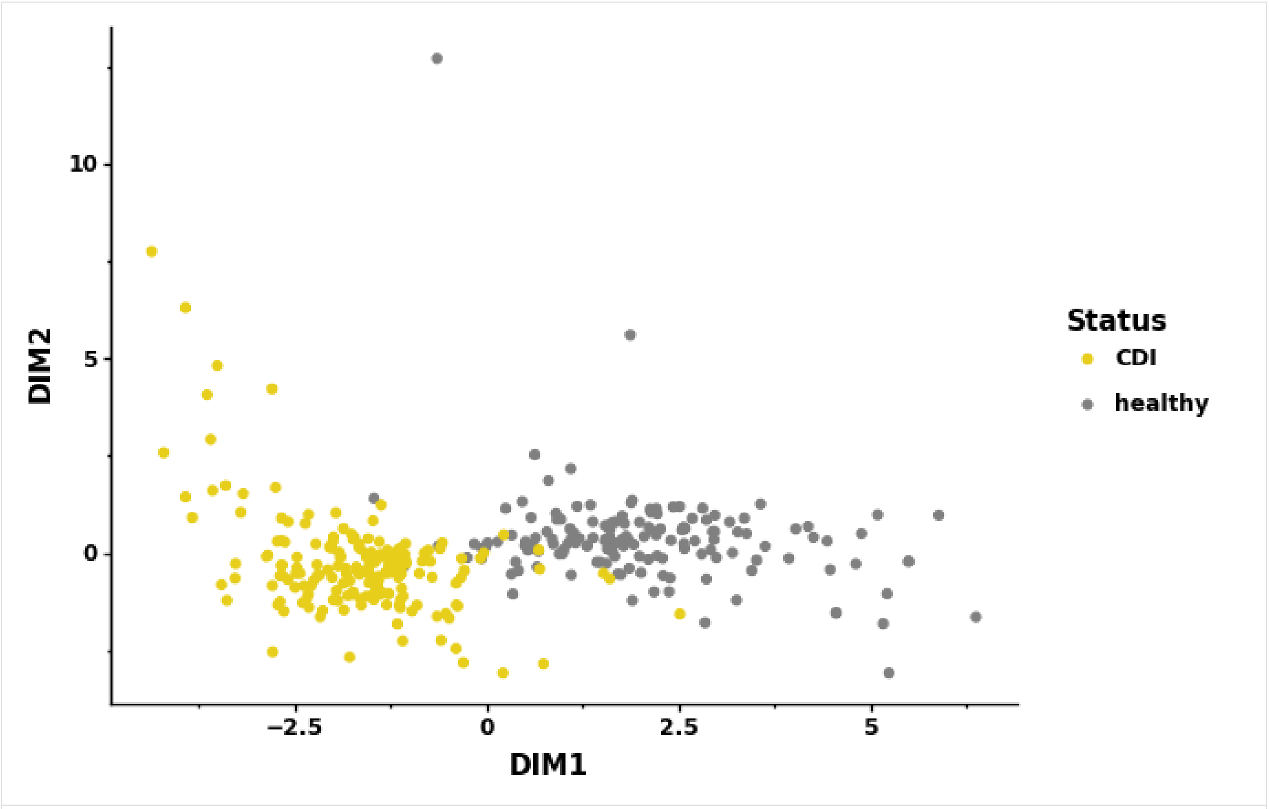


```
[38]: <ggplot: (-9223363254388601077)>
```

```
[39]: plsda_g = plsda(X_g, Y_g, labels_g)
```

```
[40]: g = ggplot(plsda_g.scores, aes(x='DIM1',y='DIM2', color='labels'))
      g += geom_point()
      g += scale_color_manual(name='Status',values = {"CDI":cdi_color, "healthy":healthy_
      ↪color})
      g += theme_classic()
      g += theme(plot_background=element_blank(),
                 panel_background=element_blank(),
                 legend_background=element_blank(),
                 axis_line=element_line(color='black'),
                 legend_text=element_text(color='black', weight='bold'),
                axis_text=element_text(color='black', weight='bold'),
                axis_title=element_text(color='black', weight='bold'),
                legend_title=element_text(color='black', weight='bold'))
      g.save('results/PLS-DA_genus.png', format='png', dpi=300, transparent=True)
      g
```

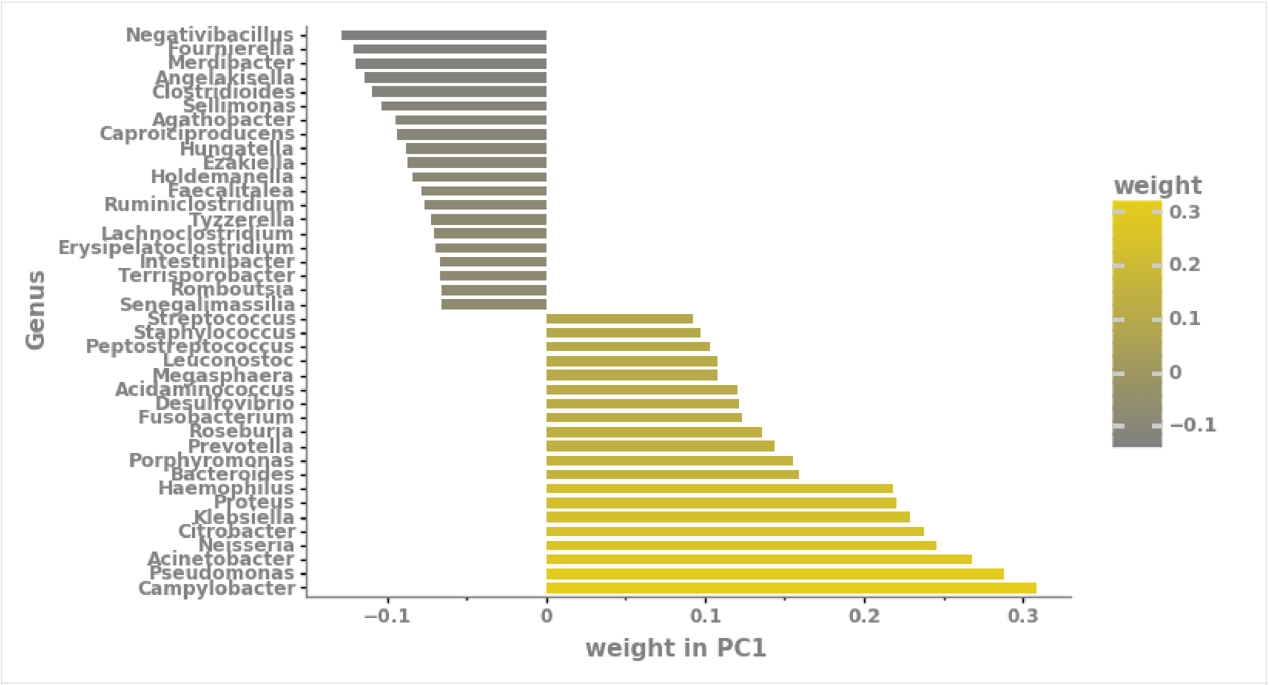
```
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:729: PlotnineWarning: Saving 6.4 x 4.8 in image.
  from_inches(height, units), units), PlotnineWarning)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:730: PlotnineWarning: Filename: results/PLS-DA_genus.png
  warn('Filename: {}'.format(filename), PlotnineWarning)
```



```
[40]: <ggplot: (8782466170155)>
```

```
[41]: g = ggplot(plsda_g.top_weights, aes(x='name',y='DIM1', fill='DIM1'))
      g += geom_bar(stat='identity', width=0.7)
      g += coord_flip()
      g += scale_fill_gradient(name = 'weight', low=healthy_color, high=cdi_color)
      g += xlab('Genus')
      g += ylab('weight in PC1')
      g += theme_classic()
      g += theme(plot_background=element_blank(),
                 panel_background=element_blank(),
                 legend_background=element_blank(),
                 axis_line=element_line(color=healthy_color),
                 legend_text=element_text(color=healthy_color, weight='bold'),
               axis_text=element_text(color=healthy_color, weight='bold'),
               axis_title=element_text(color=healthy_color, weight='bold'),
               legend_title=element_text(color=healthy_color, weight='bold'))
      g.save('results/weight_CDI_genus.png', format='png', dpi=300, transparent=True)
      g
```

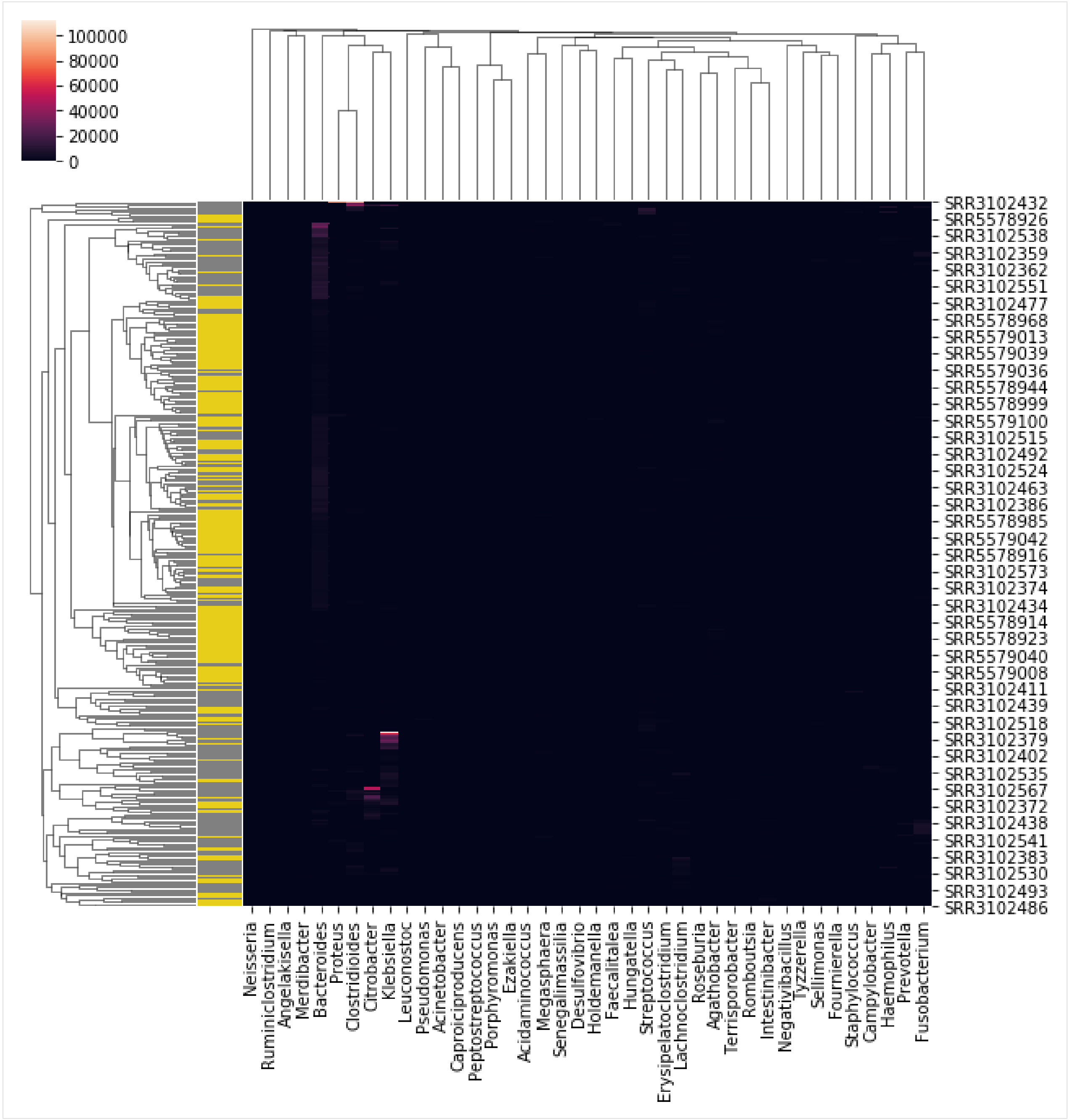
```
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:729: PlotnineWarning: Saving 6.4 x 4.8 in image.
  from_inches(height, units), units), PlotnineWarning)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:730: PlotnineWarning: Filename: results/weight_CDI_genus.png
  warn('Filename: {}'.format(filename), PlotnineWarning)
```



```
[41]: <ggplot: (8782466099411)>
```

```
[42]: X_g_heatmap = X_g.merge(labels_g, left_index=True, right_index=True).sort_values(
      ↪'labels').drop('labels', axis=1)
      X_g_heatmap.columns = ncbi.get_taxid_translator(X_g_heatmap.columns).values()
      samp_colors = list(np.where(labels_g['labels'] == 'CDI', cdi_color,healthy_color))
      sns.clustermap(X_g_heatmap.loc[:,plsda_g.top_weights['name']], row_colors=samp_colors,
      ↪ metric='braycurtis')
```

```
[42]: <seaborn.matrix.ClusterGrid at 0x7fcd3c30a198>
```



MDS

```
[43]: mds_g = mds(X_g, labels_g)
```

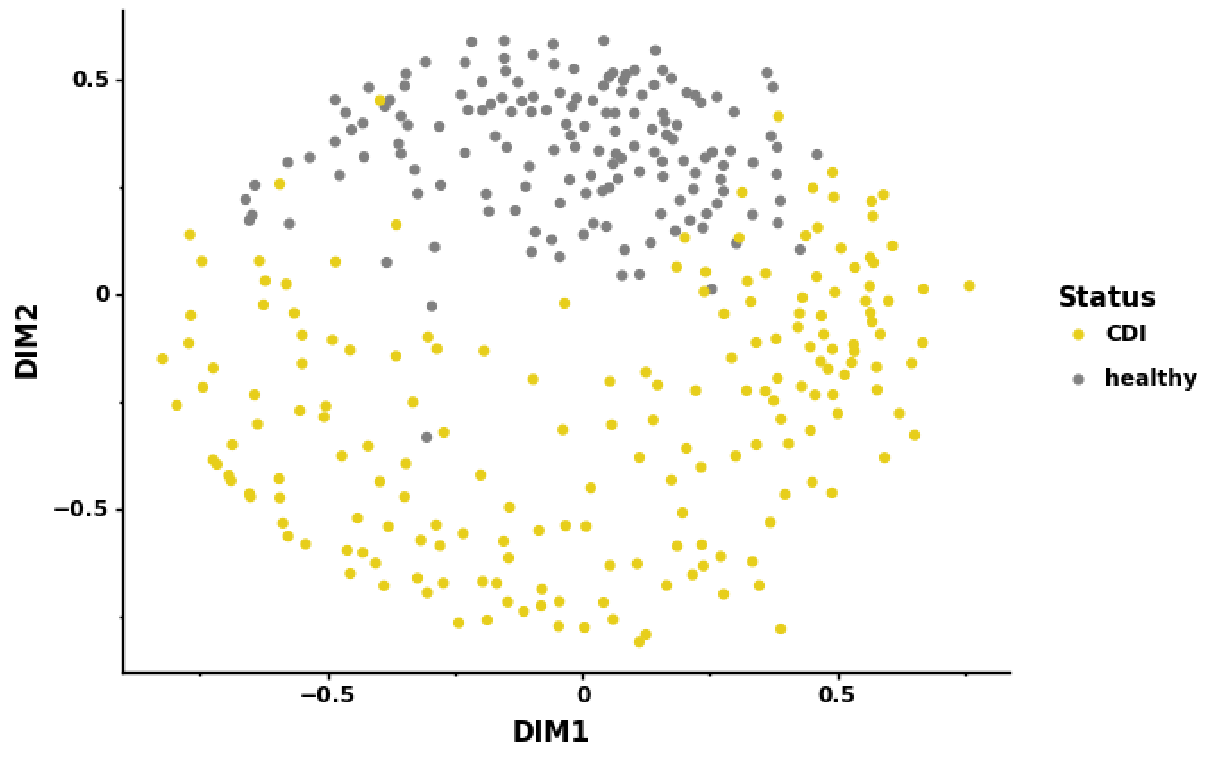
```
[44]: g = ggplot(mds_g.embedding, aes(x='DIM1',y='DIM2', color='labels'))
      g += geom_point()
      g += scale_color_manual(name='Status',values = {"CDI":cdi_color, "healthy":healthy_
      ↪color})
      g += theme_classic()
      g += theme(plot_background=element_blank(),
                 panel_background=element_blank(),
```

(continues on next page)

(continued from previous page)

```
           legend_background=element_blank(),
           axis_line=element_line(color='black'),
           legend_text=element_text(color='black', weight='bold'),
          axis_text=element_text(color='black', weight='bold'),
          axis_title=element_text(color='black', weight='bold'),
          legend_title=element_text(color='black', weight='bold'))
g.save('results/PCoA.png', format='png', dpi=300, transparent=True)
g
```

```
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:729: PlotnineWarning: Saving 6.4 x 4.8 in image.
  from_inches(height, units), units), PlotnineWarning)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:730: PlotnineWarning: Filename: results/PCoA.png
  warn('Filename: {}'.format(filename), PlotnineWarning)
```



```
[44]: <ggplot: (8782533528931)>
```

10.5.1 Pre-analysis conclusion:

Based on the clustering (Heatmap) and the MDS, there is a better separation of the two classed at the genus level.

10.5.2 Preparing data for sourcepredict

on species

[45]:
```
train_species = X_s.T.sample(frac=0.8, axis=1, random_state=2)
```

[46]:
```
test_species = X_s.T.drop(train_species.columns, axis=1)
```

[47]:
```
train_labels_species = labels_s.loc[train_species.columns,:]
```

[48]:
```
test_labels_species = labels_s.loc[test_species.columns,:]
```

[49]:
```
train_species.to_csv("source_species.csv")
test_species.to_csv("sink_species.csv")
train_labels_species.to_csv("source_labels_species.csv")
test_labels_species.to_csv("sink_labels_species.csv")
```

on genus

[50]:
```
train_genus = X_g.T.sample(frac=0.8, axis=1, random_state=2)
```

[51]:
```
test_genus = X_g.T.drop(train_genus.columns, axis=1)
```

[52]:
```
train_labels_genus = labels_g.loc[train_genus.columns,:]
```

[53]:
```
test_labels_genus = labels_g.loc[test_genus.columns,:]
```

[54]:
```
train_genus.to_csv("source_genus.csv")
test_genus.to_csv("sink_genus.csv")
train_labels_genus.to_csv("source_labels_genus.csv")
test_labels_genus.to_csv("sink_labels_genus.csv")
```

10.5.3 Running sourcepredict

on species

[55]:
```
%%time
! /projects1/users/borry/18_sourcepredict/sourcepredict -s source_species.csv -l␣
↪source_labels_species.csv sink_species.csv -n None -me tsne -di 2 -t 6 -e embedding_
↪species.csv
```

```
Step 1: Checking for unknown proportion
  == Sample: SRR5578937 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578937
                 known:97.62%
                 unknown:2.38%
```

(continues on next page)

(continued from previous page)

```
== Sample: SRR5578953 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578953
               known:97.62%
               unknown:2.38%
== Sample: SRR5578924 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578924
               known:97.62%
               unknown:2.38%
== Sample: SRR5578990 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578990
               known:97.62%
               unknown:2.38%
== Sample: SRR5579006 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579006
               known:97.62%
               unknown:2.38%
== Sample: SRR5579106 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
```

(continues on next page)

(continued from previous page)

```
      - Sample: SRR5579106
               known:97.62%
               unknown:2.38%
== Sample: SRR5579003 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579003
               known:97.62%
               unknown:2.38%
== Sample: SRR5579074 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579074
               known:97.62%
               unknown:2.38%
== Sample: SRR5578943 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578943
               known:97.62%
               unknown:2.38%
== Sample: SRR5579072 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579072
               known:97.62%
               unknown:2.38%
== Sample: SRR5579036 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
```

(continues on next page)

(continued from previous page)

```
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579036
               known:97.62%
               unknown:2.38%
== Sample: SRR5578962 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578962
               known:97.63%
               unknown:2.37%
== Sample: SRR5578931 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578931
               known:97.62%
               unknown:2.38%
== Sample: SRR5579039 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579039
               known:97.62%
               unknown:2.38%
== Sample: SRR5579061 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579061
               known:97.62%
               unknown:2.38%
== Sample: SRR5578942 ==
      Adding unknown
      Normalizing (no normalization)
```

(continues on next page)

(continued from previous page)

```
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578942
                 known:97.62%
                 unknown:2.38%
== Sample: SRR5578919 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578919
                 known:97.62%
                 unknown:2.38%
== Sample: SRR5578913 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578913
                 known:97.62%
                 unknown:2.38%
== Sample: SRR5578991 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578991
                 known:97.62%
                 unknown:2.38%
== Sample: SRR5579100 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579100
                 known:97.62%
                 unknown:2.38%
```

(continues on next page)

(continued from previous page)

```
== Sample: SRR5579037 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579037
               known:97.72%
               unknown:2.28%
== Sample: SRR5579048 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579048
               known:97.76%
               unknown:2.24%
== Sample: SRR5578952 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578952
               known:97.62%
               unknown:2.38%
== Sample: SRR5578930 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578930
               known:97.62%
               unknown:2.38%
== Sample: SRR5579017 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
```

(continues on next page)

(continued from previous page)

```
      - Sample: SRR5579017
               known:97.62%
               unknown:2.38%
== Sample: SRR5579005 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579005
               known:97.62%
               unknown:2.38%
== Sample: SRR5579055 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579055
               known:97.62%
               unknown:2.38%
== Sample: SRR5579002 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579002
               known:97.62%
               unknown:2.38%
== Sample: SRR5578968 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578968
               known:97.62%
               unknown:2.38%
== Sample: SRR5579116 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
```

(continues on next page)

(continued from previous page)

```
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579116
               known:97.62%
               unknown:2.38%
== Sample: SRR5578910 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578910
               known:97.66%
               unknown:2.34%
== Sample: SRR5578947 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578947
               known:97.62%
               unknown:2.38%
== Sample: SRR5579080 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579080
               known:97.62%
               unknown:2.38%
== Sample: SRR5579103 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579103
               known:97.62%
               unknown:2.38%
== Sample: SRR5579035 ==
      Adding unknown
      Normalizing (no normalization)
```

(continues on next page)

(continued from previous page)

```
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579035
               known:97.62%
               unknown:2.38%
== Sample: SRR5579085 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579085
               known:97.62%
               unknown:2.38%
== Sample: SRR5579079 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579079
               known:97.63%
               unknown:2.37%
== Sample: SRR5579018 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579018
               known:97.62%
               unknown:2.38%
== Sample: SRR5579001 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579001
               known:97.62%
               unknown:2.38%
```

(continues on next page)

(continued from previous page)

```
== Sample: SRR5578938 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578938
               known:97.62%
               unknown:2.38%
== Sample: SRR5579108 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579108
               known:97.62%
               unknown:2.38%
== Sample: SRR3102551 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102551
               known:97.62%
               unknown:2.38%
== Sample: SRR3102397 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102397
               known:97.62%
               unknown:2.38%
== Sample: SRR3102539 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
```

(continues on next page)

(continued from previous page)

```
      - Sample: SRR3102539
               known:96.97%
               unknown:3.03%
== Sample: SRR3102516 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102516
               known:97.62%
               unknown:2.38%
== Sample: SRR3102401 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102401
               known:97.62%
               unknown:2.38%
== Sample: SRR3102398 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102398
               known:97.62%
               unknown:2.38%
== Sample: SRR3102573 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102573
               known:97.62%
               unknown:2.38%
== Sample: SRR3102518 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
```

(continues on next page)

(continued from previous page)

```
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102518
               known:74.16%
               unknown:25.84%
== Sample: SRR3102463 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102463
               known:97.68%
               unknown:2.32%
== Sample: SRR3102581 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102581
               known:97.62%
               unknown:2.38%
== Sample: SRR3102359 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102359
               known:97.62%
               unknown:2.38%
== Sample: SRR3102427 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102427
               known:97.62%
               unknown:2.38%
== Sample: SRR3102369 ==
      Adding unknown
      Normalizing (no normalization)
```

(continues on next page)

(continued from previous page)

```
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102369
               known:97.62%
               unknown:2.38%
== Sample: SRR3102356 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102356
               known:64.17%
               unknown:35.83%
== Sample: SRR3102561 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102561
               known:97.62%
               unknown:2.38%
== Sample: SRR3102374 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102374
               known:97.62%
               unknown:2.38%
== Sample: SRR3102372 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102372
               known:97.62%
               unknown:2.38%
```

(continues on next page)

(continued from previous page)

```
  == Sample: SRR3102386 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102386
                 known:97.62%
                 unknown:2.38%
  == Sample: SRR3102486 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102486
                 known:70.98%
                 unknown:29.02%
  == Sample: SRR3102556 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102556
                 known:69.48%
                 unknown:30.52%
Step 2: Checking for source proportion
        Computing weighted_unifrac distance on species rank
         Warning: ``tree`` must be rooted.
        There is a polytomy ar the root of this taxonomic tree.
        Unifrac distances wont't  work properly.
        Computing  Bray-Curtis distance instead.

        TSNE embedding in 2 dimensions
        KNN machine learning
        Performing 5 fold cross validation on 6 cores...
        Trained KNN classifier with 10 neighbors
        -> Testing Accuracy: 0.9
        ----------------------
        - Sample: SRR5578937
                 CDI:5.35%
                 healthy:94.65%
        - Sample: SRR5578953
                 CDI:15.66%
                 healthy:84.34%
        - Sample: SRR5578924
                 CDI:11.87%
```

(continues on next page)

(continued from previous page)

```
                healthy:88.13%
        - Sample: SRR5578990
                CDI:11.32%
                healthy:88.68%
        - Sample: SRR5579006
                CDI:5.35%
                healthy:94.65%
        - Sample: SRR5579106
                CDI:5.35%
                healthy:94.65%
        - Sample: SRR5579003
                CDI:7.22%
                healthy:92.78%
        - Sample: SRR5579074
                CDI:7.25%
                healthy:92.75%
        - Sample: SRR5578943
                CDI:92.09%
                healthy:7.91%
        - Sample: SRR5579072
                CDI:5.35%
                healthy:94.65%
        - Sample: SRR5579036
                CDI:5.35%
                healthy:94.65%
        - Sample: SRR5578962
                CDI:5.35%
                healthy:94.65%
        - Sample: SRR5578931
                CDI:5.35%
                healthy:94.65%
        - Sample: SRR5579039
                CDI:15.35%
                healthy:84.65%
        - Sample: SRR5579061
                CDI:11.1%
                healthy:88.9%
        - Sample: SRR5578942
                CDI:5.35%
                healthy:94.65%
        - Sample: SRR5578919
                CDI:68.11%
                healthy:31.89%
        - Sample: SRR5578913
                CDI:8.83%
                healthy:91.17%
        - Sample: SRR5578991
                CDI:5.35%
                healthy:94.65%
        - Sample: SRR5579100
                CDI:13.42%
                healthy:86.58%
        - Sample: SRR5579037
                CDI:5.35%
                healthy:94.65%
        - Sample: SRR5579048
                CDI:5.35%
```

(continues on next page)

(continued from previous page)

```
                 healthy:94.65%
         - Sample: SRR5578952
                 CDI:14.35%
                 healthy:85.65%
         - Sample: SRR5578930
                 CDI:9.87%
                 healthy:90.13%
         - Sample: SRR5579017
                 CDI:9.84%
                 healthy:90.16%
         - Sample: SRR5579005
                 CDI:5.35%
                 healthy:94.65%
         - Sample: SRR5579055
                 CDI:9.51%
                 healthy:90.49%
         - Sample: SRR5579002
                 CDI:92.09%
                 healthy:7.91%
         - Sample: SRR5578968
                 CDI:5.35%
                 healthy:94.65%
         - Sample: SRR5579116
                 CDI:5.35%
                 healthy:94.65%
         - Sample: SRR5578910
                 CDI:8.34%
                 healthy:91.66%
         - Sample: SRR5578947
                 CDI:5.35%
                 healthy:94.65%
         - Sample: SRR5579080
                 CDI:5.35%
                 healthy:94.65%
         - Sample: SRR5579103
                 CDI:5.35%
                 healthy:94.65%
         - Sample: SRR5579035
                 CDI:9.01%
                 healthy:90.99%
         - Sample: SRR5579085
                 CDI:7.16%
                 healthy:92.84%
         - Sample: SRR5579079
                 CDI:7.01%
                 healthy:92.99%
         - Sample: SRR5579018
                 CDI:7.37%
                 healthy:92.63%
         - Sample: SRR5579001
                 CDI:5.35%
                 healthy:94.65%
         - Sample: SRR5578938
                 CDI:9.36%
                 healthy:90.64%
         - Sample: SRR5579108
                 CDI:10.13%
```

(continues on next page)

(continued from previous page)

```
                 healthy:89.87%
        - Sample: SRR3102551
                 CDI:10.39%
                 healthy:89.61%
        - Sample: SRR3102397
                 CDI:86.75%
                 healthy:13.25%
        - Sample: SRR3102539
                 CDI:88.59%
                 healthy:11.41%
        - Sample: SRR3102516
                 CDI:79.7%
                 healthy:20.3%
        - Sample: SRR3102401
                 CDI:87.61%
                 healthy:12.39%
        - Sample: SRR3102398
                 CDI:92.09%
                 healthy:7.91%
        - Sample: SRR3102573
                 CDI:92.09%
                 healthy:7.91%
        - Sample: SRR3102518
                 CDI:83.99%
                 healthy:16.01%
        - Sample: SRR3102463
                 CDI:78.19%
                 healthy:21.81%
        - Sample: SRR3102581
                 CDI:5.35%
                 healthy:94.65%
        - Sample: SRR3102359
                 CDI:92.09%
                 healthy:7.91%
        - Sample: SRR3102427
                 CDI:92.09%
                 healthy:7.91%
        - Sample: SRR3102369
                 CDI:92.09%
                 healthy:7.91%
        - Sample: SRR3102356
                 CDI:82.45%
                 healthy:17.55%
        - Sample: SRR3102561
                 CDI:78.45%
                 healthy:21.55%
        - Sample: SRR3102374
                 CDI:86.09%
                 healthy:13.91%
        - Sample: SRR3102372
                 CDI:70.12%
                 healthy:29.88%
        - Sample: SRR3102386
                 CDI:88.18%
                 healthy:11.82%
        - Sample: SRR3102486
                 CDI:86.38%
```

(continues on next page)

(continued from previous page)

```
                 healthy:13.62%
        - Sample: SRR3102556
                 CDI:88.76%
                 healthy:11.24%
Sourcepredict result written to sink_species.sourcepredict.csv
Embedding coordinates written to embedding_species.csv
CPU times: user 896 ms, sys: 320 ms, total: 1.22 s
Wall time: 1min 1s
```

on genus

[56]:
```
%%time
! /projects1/users/borry/18_sourcepredict/sourcepredict -s source_genus.csv -l source_
↪labels_genus.csv sink_genus.csv -r genus -n None -me tsne -di 2 -dt weighted_
↪unifrac -t 6 -e embedding_genus.csv
```

```
Step 1: Checking for unknown proportion
  == Sample: SRR5578953 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578953
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5578924 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578924
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5579094 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579094
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5579106 ==
        Adding unknown
```

(continues on next page)

(continued from previous page)

```
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579106
                 known:97.37%
                 unknown:2.63%
== Sample: SRR5579030 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579030
                 known:97.37%
                 unknown:2.63%
== Sample: SRR5579113 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579113
                 known:97.37%
                 unknown:2.63%
== Sample: SRR5579077 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579077
                 known:97.43%
                 unknown:2.57%
== Sample: SRR5578962 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578962
                 known:97.37%
```

(continues on next page)

(continued from previous page)

```
                unknown:2.63%
== Sample: SRR5579050 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579050
                known:97.37%
                unknown:2.63%
== Sample: SRR5578957 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578957
                known:97.37%
                unknown:2.63%
== Sample: SRR5579028 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579028
                known:97.37%
                unknown:2.63%
== Sample: SRR5579071 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579071
                known:97.38%
                unknown:2.62%
== Sample: SRR5578993 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
```

(continues on next page)

(continued from previous page)

```
      ----------------------
      - Sample: SRR5578993
               known:97.37%
               unknown:2.63%
== Sample: SRR5578963 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578963
               known:97.37%
               unknown:2.63%
== Sample: SRR5579100 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579100
               known:97.49%
               unknown:2.51%
== Sample: SRR5578906 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578906
               known:97.37%
               unknown:2.63%
== Sample: SRR5579112 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579112
               known:97.37%
               unknown:2.63%
== Sample: SRR5579013 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
```

(continues on next page)

(continued from previous page)

```
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579013
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5579042 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579042
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5578912 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578912
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5579002 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579002
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5579089 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579089
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5579087 ==
        Adding unknown
```

(continues on next page)

(continued from previous page)

```
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579087
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5579102 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579102
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5579049 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.99
        ----------------------
        - Sample: SRR5579049
                 known:98.22%
                 unknown:1.78%
  == Sample: SRR5579029 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579029
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR5578910 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578910
                 known:97.37%
```

(continues on next page)

(continued from previous page)

```
                  unknown:2.63%
== Sample: SRR5579010 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579010
                  known:97.37%
                  unknown:2.63%
== Sample: SRR5579064 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579064
                  known:97.37%
                  unknown:2.63%
== Sample: SRR5579051 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5579051
                  known:97.37%
                  unknown:2.63%
== Sample: SRR5578984 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578984
                  known:97.37%
                  unknown:2.63%
== Sample: SRR5578949 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
```

(continues on next page)

(continued from previous page)

```
      ----------------------
      - Sample: SRR5578949
               known:97.37%
               unknown:2.63%
== Sample: SRR5578975 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578975
               known:97.37%
               unknown:2.63%
== Sample: SRR5578940 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578940
               known:97.37%
               unknown:2.63%
== Sample: SRR5578925 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5578925
               known:97.41%
               unknown:2.59%
== Sample: SRR5579015 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR5579015
               known:97.37%
               unknown:2.63%
== Sample: SRR5578927 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
```

(continues on next page)

(continued from previous page)

```
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578927
                 known:97.49%
                 unknown:2.51%
  == Sample: SRR5578923 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR5578923
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR3102557 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102557
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR3102424 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.99
        ----------------------
        - Sample: SRR3102424
                 known:92.66%
                 unknown:7.34%
  == Sample: SRR3102366 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102366
                 known:93.41%
                 unknown:6.59%
  == Sample: SRR3102422 ==
        Adding unknown
```

(continues on next page)

(continued from previous page)

```
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 0.99
      ----------------------
      - Sample: SRR3102422
               known:89.06%
               unknown:10.94%
== Sample: SRR3102462 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 0.98
      ----------------------
      - Sample: SRR3102462
               known:69.99%
               unknown:30.01%
== Sample: SRR3102449 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102449
               known:96.44%
               unknown:3.56%
== Sample: SRR3102526 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102526
               known:96.96%
               unknown:3.04%
== Sample: SRR3102497 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102497
               known:95.78%
```

(continues on next page)

(continued from previous page)

```
                unknown:4.22%
== Sample: SRR3102402 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102402
                known:97.37%
                unknown:2.63%
== Sample: SRR3102440 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102440
                known:97.37%
                unknown:2.63%
== Sample: SRR3102463 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102463
                known:97.37%
                unknown:2.63%
== Sample: SRR3102379 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102379
                known:91.82%
                unknown:8.18%
== Sample: SRR3102529 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
```

(continues on next page)

(continued from previous page)

```
      ----------------------
      - Sample: SRR3102529
               known:91.89%
               unknown:8.11%
== Sample: SRR3102427 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102427
               known:97.39%
               unknown:2.61%
== Sample: SRR3102550 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102550
               known:97.37%
               unknown:2.63%
== Sample: SRR3102410 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 0.99
      ----------------------
      - Sample: SRR3102410
               known:94.87%
               unknown:5.13%
== Sample: SRR3102376 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
      KNN machine learning
      Training KNN classifier on 6 cores...
      -> Testing Accuracy: 1.0
      ----------------------
      - Sample: SRR3102376
               known:92.88%
               unknown:7.12%
== Sample: SRR3102533 ==
      Adding unknown
      Normalizing (no normalization)
      Computing Bray-Curtis distance
      Performing MDS embedding in 2 dimensions
```

(continues on next page)

(continued from previous page)

```
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102533
                 known:97.39%
                 unknown:2.61%
  == Sample: SRR3102489 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102489
                 known:95.74%
                 unknown:4.26%
  == Sample: SRR3102490 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102490
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR3102580 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102580
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR3102375 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102375
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR3102483 ==
        Adding unknown
```

(continues on next page)

(continued from previous page)

```
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102483
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR3102535 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102535
                 known:96.31%
                 unknown:3.69%
  == Sample: SRR3102446 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102446
                 known:97.41%
                 unknown:2.59%
  == Sample: SRR3102527 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102527
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR3102409 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102409
                 known:97.37%
```

(continues on next page)

(continued from previous page)

```
                 unknown:2.63%
  == Sample: SRR3102362 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102362
                 known:97.37%
                 unknown:2.63%
  == Sample: SRR3102487 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 0.99
        ----------------------
        - Sample: SRR3102487
                 known:96.66%
                 unknown:3.34%
  == Sample: SRR3102517 ==
        Adding unknown
        Normalizing (no normalization)
        Computing Bray-Curtis distance
        Performing MDS embedding in 2 dimensions
        KNN machine learning
        Training KNN classifier on 6 cores...
        -> Testing Accuracy: 1.0
        ----------------------
        - Sample: SRR3102517
                 known:97.37%
                 unknown:2.63%
Step 2: Checking for source proportion
        Computing weighted_unifrac distance on genus rank
        TSNE embedding in 2 dimensions
        KNN machine learning
        Performing 5 fold cross validation on 6 cores...
        Trained KNN classifier with 10 neighbors
        -> Testing Accuracy: 0.85
        ----------------------
        - Sample: SRR5578953
                 CDI:2.9%
                 healthy:97.1%
        - Sample: SRR5578924
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579094
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579106
                 CDI:1.71%
                 healthy:98.29%
```

(continues on next page)

(continued from previous page)

```
        - Sample: SRR5579030
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579113
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579077
                 CDI:13.41%
                 healthy:86.59%
        - Sample: SRR5578962
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579050
                 CDI:2.26%
                 healthy:97.74%
        - Sample: SRR5578957
                 CDI:2.82%
                 healthy:97.18%
        - Sample: SRR5579028
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579071
                 CDI:2.77%
                 healthy:97.23%
        - Sample: SRR5578993
                 CDI:2.78%
                 healthy:97.22%
        - Sample: SRR5578963
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579100
                 CDI:6.98%
                 healthy:93.02%
        - Sample: SRR5578906
                 CDI:3.1%
                 healthy:96.9%
        - Sample: SRR5579112
                 CDI:2.95%
                 healthy:97.05%
        - Sample: SRR5579013
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579042
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5578912
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR5579002
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579089
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579087
                 CDI:88.23%
                 healthy:11.77%
```

(continues on next page)

(continued from previous page)

```
        - Sample: SRR5579102
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579049
                 CDI:9.53%
                 healthy:90.47%
        - Sample: SRR5579029
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5578910
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5579010
                 CDI:13.19%
                 healthy:86.81%
        - Sample: SRR5579064
                 CDI:2.48%
                 healthy:97.52%
        - Sample: SRR5579051
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5578984
                 CDI:44.06%
                 healthy:55.94%
        - Sample: SRR5578949
                 CDI:3.11%
                 healthy:96.89%
        - Sample: SRR5578975
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5578940
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5578925
                 CDI:2.55%
                 healthy:97.45%
        - Sample: SRR5579015
                 CDI:1.71%
                 healthy:98.29%
        - Sample: SRR5578927
                 CDI:2.93%
                 healthy:97.07%
        - Sample: SRR5578923
                 CDI:4.14%
                 healthy:95.86%
        - Sample: SRR3102557
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102424
                 CDI:92.07%
                 healthy:7.93%
        - Sample: SRR3102366
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102422
                 CDI:90.09%
                 healthy:9.91%
```

(continues on next page)

(continued from previous page)

```
        - Sample: SRR3102462
                 CDI:93.16%
                 healthy:6.84%
        - Sample: SRR3102449
                 CDI:92.87%
                 healthy:7.13%
        - Sample: SRR3102526
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102497
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102402
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102440
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102463
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102379
                 CDI:94.18%
                 healthy:5.82%
        - Sample: SRR3102529
                 CDI:93.06%
                 healthy:6.94%
        - Sample: SRR3102427
                 CDI:55.04%
                 healthy:44.96%
        - Sample: SRR3102550
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102410
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102376
                 CDI:2.71%
                 healthy:97.29%
        - Sample: SRR3102533
                 CDI:93.57%
                 healthy:6.43%
        - Sample: SRR3102489
                 CDI:75.7%
                 healthy:24.3%
        - Sample: SRR3102490
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102580
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102375
                 CDI:95.31%
                 healthy:4.69%
        - Sample: SRR3102483
                 CDI:45.14%
                 healthy:54.86%
```

(continues on next page)

(continued from previous page)

```
        - Sample: SRR3102535
                CDI:92.53%
                healthy:7.47%
        - Sample: SRR3102446
                CDI:95.31%
                healthy:4.69%
        - Sample: SRR3102527
                CDI:95.31%
                healthy:4.69%
        - Sample: SRR3102409
                CDI:95.31%
                healthy:4.69%
        - Sample: SRR3102362
                CDI:95.31%
                healthy:4.69%
        - Sample: SRR3102487
                CDI:95.31%
                healthy:4.69%
        - Sample: SRR3102517
                CDI:95.31%
                healthy:4.69%
Sourcepredict result written to sink_genus.sourcepredict.csv
Embedding coordinates written to embedding_genus.csv
CPU times: user 1.18 s, sys: 368 ms, total: 1.55 s
Wall time: 1min 19s
```

10.6 Reading Sourcepredict results

[57]:
```
from sklearn.metrics import accuracy_score
```

[58]:
```
pred_genus = pd.read_csv("sink_genus.sourcepredict.csv", index_col=0)
test_labels_genus = pd.read_csv("sink_labels_genus.csv", index_col=0)
```

[59]:
```
conf_table_genus = pred_genus.idxmax(axis=0).to_frame(name='predicted').merge(test_
↪labels_genus, left_index=True, right_index=True)
```

[60]:
```
conf_table_genus = conf_table_genus.dropna()
```

[61]:
```
conf_table_genus.shape
```

[61]:
```
(68, 2)
```

[62]:
```
conf_table_genus.apply(pd.value_counts, axis=0)
```

[62]:
```
         predicted  labels
healthy         38      38
CDI             30      30
```

[63]:
```
acc_genus = accuracy_score(y_true=conf_table_genus['labels'], y_pred=conf_table_genus[
↪'predicted'])
print(f"Accuracy: {round(acc_genus,2)}")
```

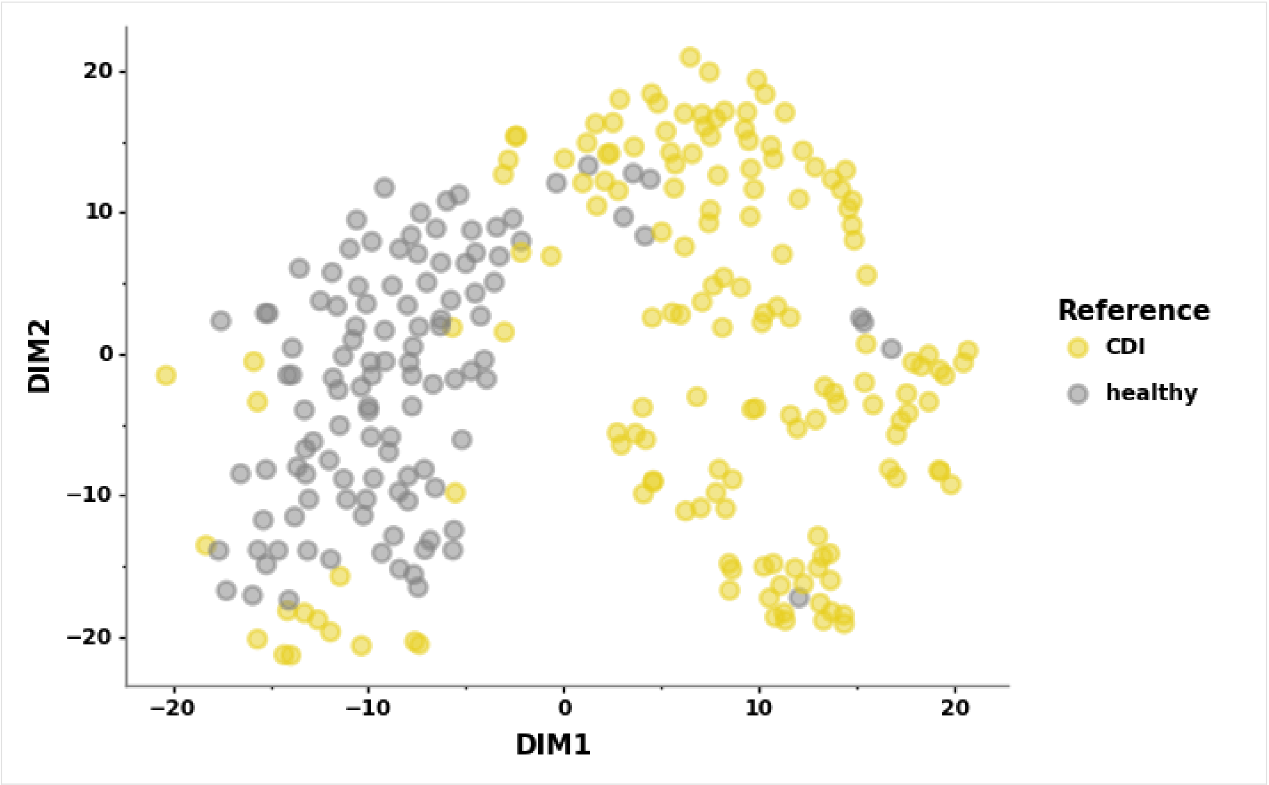
```
Accuracy: 0.94
```

[64]:
```
from plotnine import *
```

[65]:
```
embed = pd.read_csv("embedding_genus.csv", index_col=0)
embed = embed.rename(columns={'labels':'type'})
embed['type'] = embed['type'].str.replace('CDI','source').replace('healthy','source')
embed = embed.join(labels_g['labels']).rename(columns={'labels':'actual'})
embed = embed.join(pd.Series(pred_genus.idxmax(), name='predicted'))
```

[66]:
```
g = ggplot(embed.query("type == 'source'"), aes(x='PC1',y='PC2', color='labels'))
g += geom_point(size=3, stroke=1, alpha=0.5)
g += scale_color_manual(name = 'Reference', values = {"CDI":cdi_color, "healthy":
↪healthy_color})
g += xlab('DIM1')
g += ylab('DIM2')
g += theme_classic()
g += theme(plot_background=element_blank(),
           panel_background=element_blank(),
           legend_background=element_blank(),
           axis_line=element_line(color=healthy_color),
           legend_text=element_text(color='black', weight='bold'),
          axis_text=element_text(color='black', weight='bold'),
          axis_title=element_text(color='black', weight='bold'),
          legend_title=element_text(color='black', weight='bold'))
g.save('results/train_embedding.png', format='png', dpi=300, transparent=True)
g
```

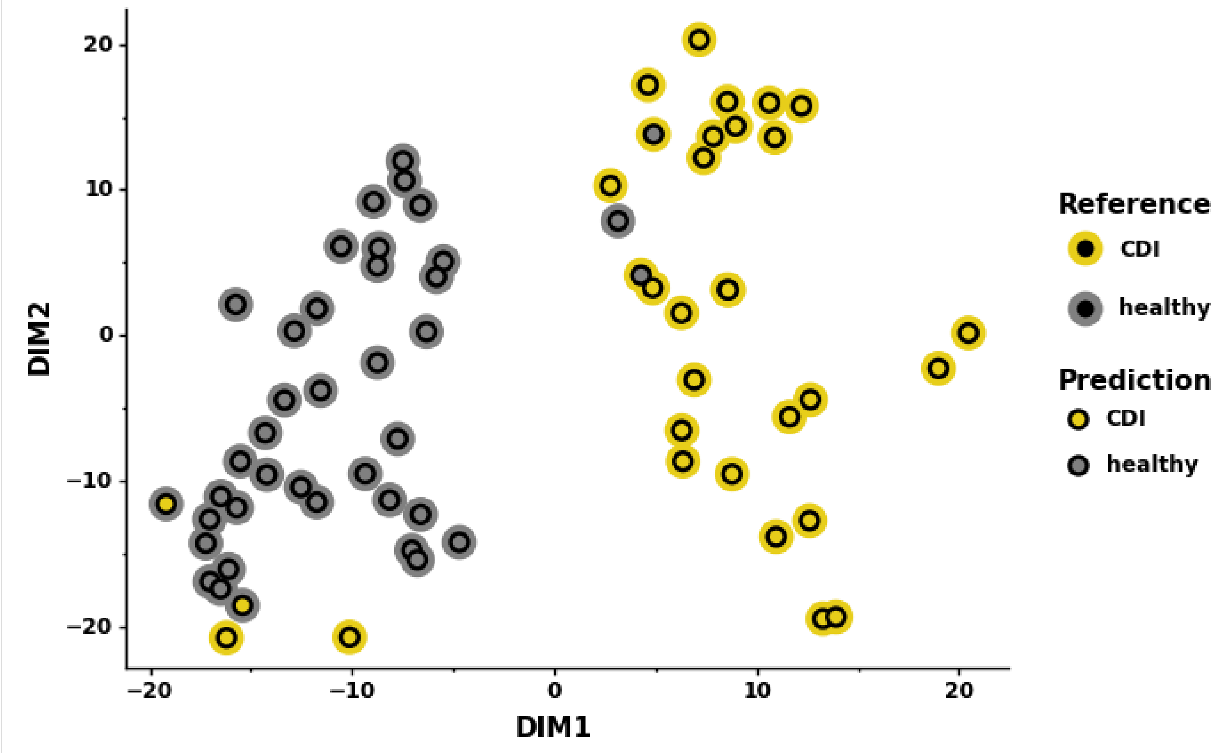
```
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:729: PlotnineWarning: Saving 6.4 x 4.8 in image.
  from_inches(height, units), units), PlotnineWarning)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:730: PlotnineWarning: Filename: results/train_embedding.png
  warn('Filename: {}'.format(filename), PlotnineWarning)
```



```
[66]: <ggplot: (8782530729613)>
```

```
[67]: g = ggplot(embed.query("type == 'sink'"), aes(x='PC1',y='PC2', color='predicted'))
g += geom_point(size=4, shape='o', fill = 'black', stroke=2)
g += geom_point(data=embed.query("type == 'sink'"), mapping=aes(x='PC1',y='PC2', fill=
↪'actual'), size=3, color='black', stroke=1)
g += scale_color_manual(name = 'Reference', values = {"CDI":cdi_color, "healthy":
↪healthy_color})
g += scale_fill_manual(name = 'Prediction', values = {"CDI":cdi_color, "healthy":
↪healthy_color})
g += xlab('DIM1')
g += ylab('DIM2')
g += theme_classic()
g += theme(plot_background=element_blank(),
           panel_background=element_blank(),
           legend_background=element_blank(),
           axis_line=element_line(color='black'),
           legend_text=element_text(color='black', weight='bold'),
          axis_text=element_text(color='black', weight='bold'),
          axis_title=element_text(color='black', weight='bold'),
          legend_title=element_text(color='black', weight='bold'))
g.save('results/test_embedding.png', format='png', dpi=300, transparent=True)
g
```

```
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:729: PlotnineWarning: Saving 6.4 x 4.8 in image.
  from_inches(height, units), units), PlotnineWarning)
/projects1/users/borry/15_miniconda3/envs/sourcepredict/lib/python3.6/site-packages/
↪plotnine/ggplot.py:730: PlotnineWarning: Filename: results/test_embedding.png
  warn('Filename: {}'.format(filename), PlotnineWarning)
```



20
10
0
−10
−20
DIM2
−20
−10
0
10
20
DIM1
Reference
CDI
healthy
Prediction
CDI
healthy

```
[67]: <ggplot: (-9223363254324085430)>
```

CHAPTER 11

Comparing with Sourcetracker 2

Generating the data for Sourcetracker 2

```
[68]: X_g.T.to_csv("st_genus.txt", sep="\t", index_label="TAXID")
```

test_labels_genus['SourceSink']= ['sink']*test_labels_genus.shape[0]

```
[69]: train_labels_genus['SourceSink'] = ['source']*train_labels_genus.shape[0]
```

```
[70]: st_metadata = train_labels_genus.append(test_labels_genus).rename(columns = {"labels":
      ↪"Env"})[['SourceSink','Env']]
      st_metadata['SourceSink'][train_labels_genus.index] = ['source']*train_labels_genus.
      ↪shape[0]
      st_metadata['SourceSink'][test_labels_genus.index] = ['sink']*test_labels_genus.
      ↪shape[0]
      st_metadata
```

```
[70]:            SourceSink      Env
      SRR5579101     source  healthy
      SRR3102473     source      CDI
      SRR3102501     source      CDI
      SRR5578964     source  healthy
      SRR5579017     source  healthy
      ...               ...      ...
      SRR3102527       sink      CDI
      SRR3102409       sink      CDI
      SRR3102362       sink      CDI
      SRR3102487       sink      CDI
      SRR3102517       sink      CDI

      [340 rows x 2 columns]
```

```
[71]: st_metadata.to_csv("st_genus_metadata.csv", sep="\t", index_label='#SampleID')
```

```
sourcetracker2 gibbs -i st_genus.biom -m st_genus_metadata.csv -o st_genus_out --jobs␣
↪6 --source_rarefaction_depth 0 --sink_rarefaction_depth 0
```

Reading sourcetracker results

```
[72]: st_pred = pd.read_csv("st_genus_out/mixing_proportions.txt", sep = "\t", index_col=0)
```

```
[73]: st2_pred = st_pred.idxmax(axis=1).to_frame(name='predicted').merge(test_labels_genus,␣
↪left_index=True, right_index=True).rename(columns={'labels':'actual'})
```

```
[74]: st2_pred.head()
```

```
[74]:            predicted   actual
SRR5578953   healthy  healthy
SRR5578924   healthy  healthy
SRR5579094   healthy  healthy
SRR5579106   healthy  healthy
SRR5579030   healthy  healthy
```

```
[75]: st_acc_genus = accuracy_score(y_true=st2_pred['actual'], y_pred=st2_pred['predicted'])
print(f"Accuracy: {round(st_acc_genus,2)}")
```

```
Accuracy: 0.8
```

CHAPTER 12

Indices and tables

- genindex
- modindex
- search