MetaBin: a package for calculating the alignment-free sequence comparison measures d2S and d2* based on binned metagenomic data sets

Version: 1.1

Authors: Kai Song, Jie Ren, Fengzhu Sun

Maintainer: Kai Song ksong@qdu.edu.cn

Description

The package provides functions to bin the next-generation sequencing reads in paired-end or single-end fastq files and calculate the alignment-free sequence comparison measures d_2^S and d_2^* between the binned metagenomic data sets. We show that d_2^S and d_2^* based on binned data are markedly better than the original version defined in Jiang et al. 2012 [1]. We propose to bin based on sequence signatures (k-tuple word occurrence). In particular, several Markov models have been trained for different groups of bacterial genomes. A read is then binned to a group under which the sequence has the highest likelihood. Once the reads are binned, d_2^S and d_2^* are computed based on the binned reads, i.e. the expected k-tuple frequencies are the weighted sum of the expectation in each bin.

Thus, the package provides functions for

(1) Counting the number of occurrences of k-tuples for each reads

(2) Computing the likelihood of each read based on k-tuple count under each of the Markov models

(3) Compute d_2^S and d_2^* based on the binned reads.

[1] *Jiang, Bai, et al. "Comparison of metagenomic samples using sequence signatures." BMC Genomics 13.1 (2012): 730.*

Dependencies

The python (>=2.7) and R (>=3.5) are needed for the MetaBin execution.

Installation

To quick start, first download the package file MetaBin.tar.bz2 according to your operating system.

For Linux user, you can install the package from the command line. Simply type the following to the command line,

```
tar jxvf MetaBin.tar.bz2
```

Usage

(1) Calculation of d_2^S and d_2^* based on binned reads using paired-end NGS data.

First, one can download the four trained Markov models of order 9 from "/trained_model/", then decompress these files using:

```
gunzip Trained_Markov_Model1_order9.txt.gz
```

then, put the four files of trained Markov models in the directory:

```
<path_to_the_MetaBin>/MetaBin/Trained_Models/
```

As an example, the package provides two testing data sets containing 10,000 paired-end metagenomic reads in the directory of "/test_data/". The usage is:

```
python MetaBin_paired-end_1.1.py ./test_data/test_sample1_1.fq ./test_data/test_sample1_2.fq ./test_data/test_sample2_1.fq ./test_data/test_sample2_2.fq
```

There are four inputs in the command: the first two inputs are the paths of the first paired-end sample, and the last two inputs are the paths of the second paired-end sample.

(2) Calculation of d_2^S and d_2^* based on binned reads using single-end NGS data.

For single-end samples, one can use the package of MetaBin_single-end_1.1.py as follows:

```
python MetaBin_paired-end_1.1.py ./test_data/test_sample1_1.fq ./test_data/test_sample2_1.fq
```

There are two inputs in the command: the first is the path of the first single-end sample and the second is the path of the second sample.

The above two packages only support the input data sets with format of “fastq”. The trained Markov models used above were the four files the package provided or other four files user trained with the same file names as the package provided.

The result will be something like the following.

```
ktuple length = 6, Markov order = 0, d2* = 0.0592623964171612, d2S = 0.0915650304746318
ktuple length = 6, Markov order = 1, d2* = 0.0574278561214909, d2S = 0.0882955676376168
ktuple length = 6, Markov order = 2, d2* = 0.0637138897094841, d2S = 0.10823670694633
ktuple length = 6, Markov order = 3, d2* = 0.117116135764618, d2S = 0.17260288427926
ktuple length = 6, Markov order = 4, d2* = 0.136438578157252, d2S = 0.208822473467341
ktuple length = 7, Markov order = 0, d2* = 0.0680981880711538, d2S = 0.101735520292058
ktuple length = 7, Markov order = 1, d2* = 0.0691245490538536, d2S = 0.106678043300889
ktuple length = 7, Markov order = 2, d2* = 0.0799309571914977, d2S = 0.129357141053496
ktuple length = 7, Markov order = 3, d2* = 0.133040889193497, d2S = 0.193118522392043
ktuple length = 7, Markov order = 4, d2* = 0.152565313633223, d2S = 0.219368157033875
ktuple length = 8, Markov order = 0, d2* = 0.0856694341802808, d2S = 0.122030588995556
ktuple length = 8, Markov order = 1, d2* = 0.0930350586207397, d2S = 0.136564400819346
ktuple length = 8, Markov order = 2, d2* = 0.116942640936863, d2S = 0.170288772457853
ktuple length = 8, Markov order = 3, d2* = 0.183004365476347, d2S = 0.238424736319805
ktuple length = 8, Markov order = 4, d2* = 0.21576307720542, d2S = 0.27173519203698
ktuple length = 9, Markov order = 0, d2* = 0.126241401495429, d2S = 0.168756494781778
ktuple length = 9, Markov order = 1, d2* = 0.146889646782306, d2S = 0.195862930211969
ktuple length = 9, Markov order = 2, d2* = 0.19088221026967, d2S = 0.241179702833416
ktuple length = 9, Markov order = 3, d2* = 0.264002461732686, d2S = 0.30710789260902
```

The first column is the length of k-tuple, the second column is the Markov order used for background sequences, the third and fourth columns are d2* and d2S values.

(3) Training Markov models using users' database

A function is also added to the package to allow users to train the Markov model using their own database for bacterial genomic sequences.

To start with, one fasta formated files, containing bacterial sequences, need to be specified. The directory where the file of the trained model will be saved and the name of the model need to be set as well.

The usage is:

python Markov_Construction.py <input-path>/input.fa <output-path>/Markov_model_order9.txt

For example, one can use the test fasta file "test-sequences.fasta" in the directory of "./test_data/" as following:

python Markov_Construction.py ./test_data/input-sequences.fasta ./Trained_Models/input-sequences.Markov_model.order9.txt

There are two inputs in the command. The first is the path of the fasta file used for Markov model construction. The second is the path of the output for the trained Markov model.