bioOTU: an improved method for simultaneous taxonomic assignments and OTUs clustering of 16s rRNA gene sequences (**Supplementary Data**)

Shi-Yi Chen *et al.*

1. Protocol of OTUs annotation at species level

To taxonomically annotate OTUs at species level, we provide a brief protocol in connection with the results as outputted from bioOTU, including the preparation of reference database, extraction of representative sequences from OTUs, and homologous search against reference database using the combination of TaxCollector (http://github.com/audy/taxcollector.git), USEARCH (http://drive5.com/usearch) and our custom Python scripts (http://chenshiyi.com/biootu.html).

Preparation of reference database

(1) We download RDP database (RDP Release 11.4) of bacterial 16S rRNA sequences in the unaligned format by visiting URL (http://rdp.cme.msu.edu/misc/resources.jsp), or by the following terminal command:

```
$ wget http://rdp.cme.msu.edu/download/current_Bacteria_unaligned.fa.gz
$ gunzip current_Bacteria_unaligned.fa.gz
```

(2) For the downloaded database, the taxonomic information must be first curated for only retaining the sequences which are successfully annotated at species level. Two external files (“names.dmp” and “nodes.dmp”) containing taxonomy information are retrieved from NCBI ftp site (ftp://ftp.ncbi.nih.gov/pub/taxonomy).

```
$ wget ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz
$ gunzip | tar -xvf taxdump.tar.gz
$ python taxcollector.py names.dmp nodes.dmp current_Bacteria_unaligned.fa \
> current_Bacteria_unaligned_specie.fa
```

(3) To further discard reference sequences failing to be strictly annotated at species level (such as “GENUS_sp.”) by custom script:

```
$ python reference_filter.py --input current_Bacteria_unaligned_specie.fa
```

(4) To avoid potential bias on the taxonomic annotation due to differential copy

number of reference species included in the present database, we therefore dereplicate the redundant records with 100% identical sequence under same species.

```
$ python dereplication_of_speies_database.py --input \
current_Bacteria_unaligned_specie_sp.fa --processors 8
```

Representative sequences of OTUs

(5) Because bioOTU employs two methods (both the taxonomy-guided and heuristic search) for OTUs clustering, we also select their representative sequences by different standards. For the OTUs as constructed by taxonomy-guided clustering method, we first extract these successfully assigned sequences by RDP classifier at genus level from each OTU and sort them in a descending order by indices of both sample size and abundance. Subsequently, we can select the representative sequences by specifying the maximum number for outputting and minimum threshold of sample size using the custom script:

```
$ python represent_sequence_get.py –outlist taxonomy_guided_OTU.list        \
--fasta taxonomy_guided_OTU.fa --number 10 --sample_size 1
```

(6) For OTUs as constructed by heuristic clustering method, we select member having the highest sample size and abundance as the representative sequence for each OTU. If more sequences than one meet this criterion, one of them is randomly selected. This file, named "*heuristic_search_represent.fasta*", has already been outputted from bioOTU. Here, we merge the both files of representative sequences together as below:

```
$ cat taxonomy-guided_represent.fasta heuristic_search_represent.fasta    \
> total_represent_sequence.fasta
```

Homologous search against reference database

(7) We employ the efficient tool of USEARCH (usearch8.0.1623) for homologous search against reference database by the global alignment algorithm (-usearch_global). All satisfactory hits for each query are outputted, and during which the parameters of sequence identity (-id) and query coverage by alignment (-query_cov) should be

specified.

```
$ usearch-usearch_global total_represent_sequence.fasta        \
  –db current_Bacteria_unaligned_11.4_used_for_specie_annotataion.fasta        \
–id 0.97 -query_cov 0.98 –userfields query+target+id+qcov        \
–maxaccepts 0 –strand both –userout all_OTU_represent_usaerch_out.txt
```

Consensus species of OTUs

(8) Here, we propose a new algorithm for determining the consensus species of OTUs annotation based on the results of homologous search. Briefly, we first subdivide the index of sequence identity into different grades: grade 1 (0.99 ≤ identity ≤ 1.0), grade 2 (0.98 ≤ identity < 0.99), grade 3 (0.97 ≤ identity < 0.98), and so on (it will end at the minimum search threshold of identity). Therefore, all returned target sequences from homologous search can be positioned into different grades according to their values of alignment identity. For each OTU, only these target sequences positioned in the top grade (with higher identity) are selected and used for determining the most frequently observed species, which is herein taken as the consensus species for this OTU. This process is performed by custom script:

```
$ python annotated.py --i all_OTU_represent_usaerch_out.txt
```

2. Definition of NMI score

The normalized mutual information (NMI) score is a widely used surrogate for evaluating the clustering results of 16s rRNA genes sequences, which is computed by comparing the ground truth (golden standard) with clustering profiles. Here, we suppose that ***S*** sequences are classified into ***M*** annotated species ($\Omega = \{s_1, s_2, ..., s_M\}$) and ***N*** constructed OTUs ($\Pi = \{c_1, c_2, ..., c_N\}$).The NMI between two datasets of both the annotated species and constructed OTUs is defined as

$$NMI(\Omega, \Pi) = \frac{I(\Omega;\Pi)}{\left[H(\Omega) + H(\Pi)\right]/2},$$

where I is the mutual information and H is the entropy. And $I(\Omega;\Pi)$, $H(\Omega)$ and $H(\Pi)$ are further defined as

$$I(\Omega;\Pi)=\sum_i\sum_j P(s_i\cap c_j)\log\frac{P(s_i\cap c_j)}{P(s_i)P(c_j)},$$

$$H(\Omega)=-\sum_i P(s_i)\log P(s_i),$$

$$H(\Pi)=-\sum_j P(c_j)\log P(c_j),$$

where $P(s_i)$, $P(c_j)$, and $P(s_i\cap c_j)$ are the probabilities of a sequence being in species s_i, OTU c_j, and the intersection of s_i and c_j, respectively. k = 1, 2, …, M; j = 1, 2, …, N.

Here, we also provide a Python script to calculate NMI score, which requires the input of one tab-delimited file containing three columns. For this input file, the three columns are the name of sequences, name of reference species (gold standard), and label of the corresponding OTUs, respectively. This script is executed as below and also freely available at: http://chenshiyi.com/biootu.html.

```
$ python NMI_calculation.py --input <tab_delimited_file>
```

Reference:

1. Manning, C.D., Raghavan, P. and Schütze, H. (2009) An introduction to information retrieval. Cambridge University Press, Cambridge, England.

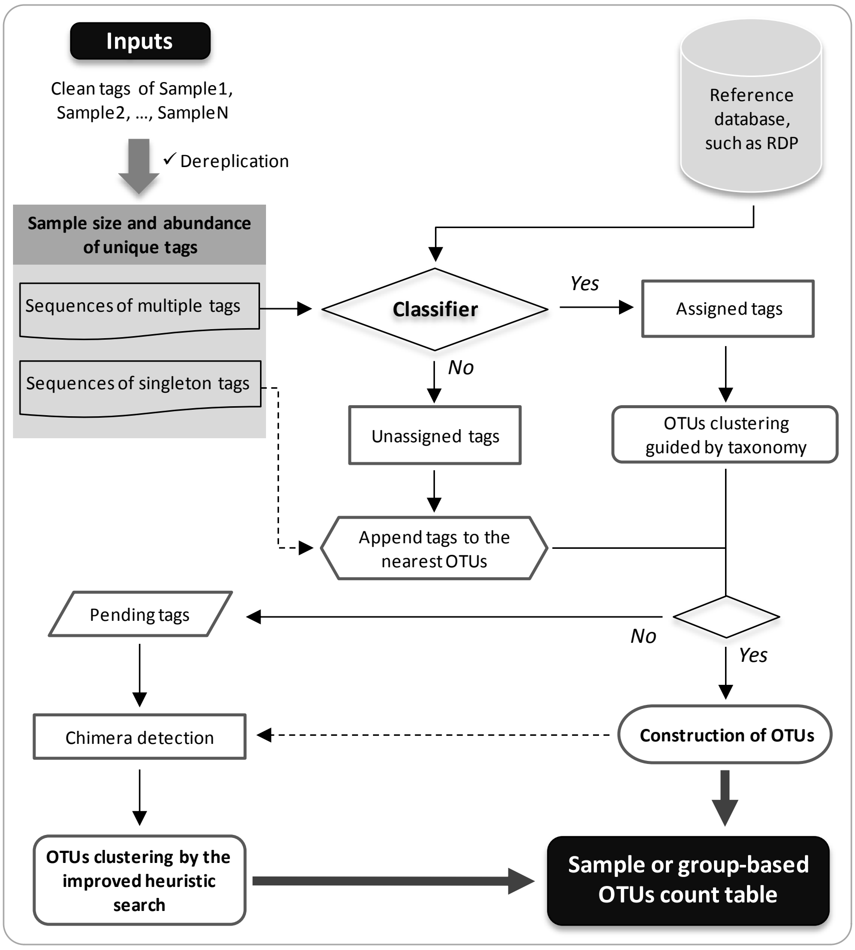
Supplementary Table S1: Dataset of the retrieved mock community consisting of 21 known organisms

NCBI SRA accessions		Region	Samples	Raw tags	Clean tags
SRR record	**SRX record**				
SRR053860	SRX021562	V3-V5R	EV1	14, 172	2, 040
SRR053854	SRX021561	V3-V5R	EV2	12, 875	1, 610
SRR053848	SRX021560	V3-V5R	EV3	14, 381	1, 931
SRR053842	SRX021559	V3-V5R	EV4	11, 463	1, 537
SRR053836	SRX021558	V3-V5R	EV5	12, 724	1, 810
SRR053704	SRX021536	V3-V5R	ST1	9, 977	1, 151
SRR053710	SRX021537	V3-V5R	ST2	6, 298	711
SRR053716	SRX021538	V3-V5R	ST3	12, 187	1, 524
SRR053722	SRX021539	V3-V5R	ST4	21, 022	1, 959
SRR053728	SRX021540	V3-V5R	ST5	28, 465	2, 217
Total	—	—	—	143, 564	16, 490

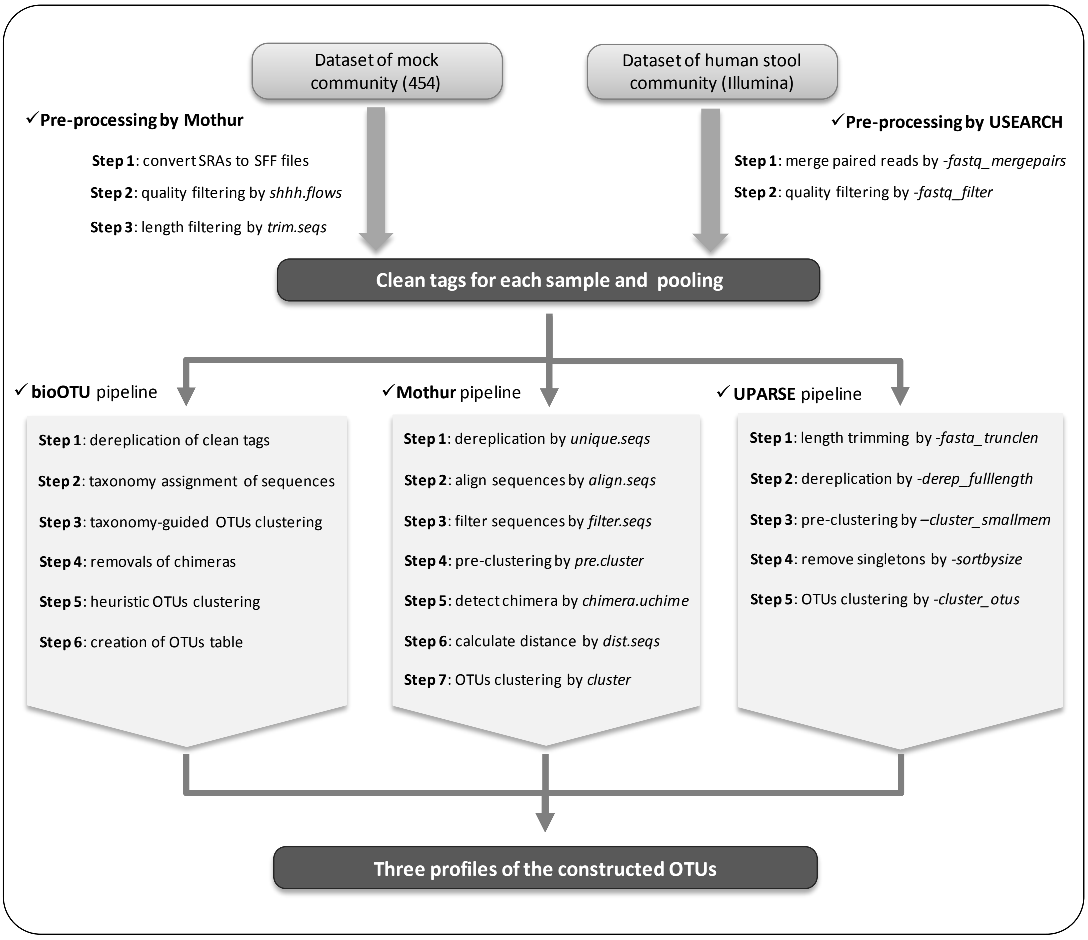
Samples of EV1~5 and ST1~5 are even and staggered mixtures, respectively.

Supplementary table 2: Dataset of the retrieved real community of human stool

NCBI SRA accession number		**Region**	**Samples**	**Raw paired reads**	**Clean tags**
SRR record	**SRX record**				
SRR1273404	SRX534396	V3	Healthy1	302, 896	295, 014
SRR1273403	SRX534395	V3	Healthy2	216, 036	211, 466
SRR1273402	SRX534394	V3	Healthy3	545, 201	528, 462
SRR1273401	SRX534393	V3	Healthy4	370, 630	361, 204
SRR1273398	SRX534390	V3	Healthy5	1, 277, 173	1, 234, 712
SRR1273390	SRX534383	V3	Healthy6	430, 006	419, 548
SRR1273389	SRX534382	V3	Healthy7	430, 663	423, 122
SRR1273385	SRX534381	V3	Healthy8	288, 119	280, 815
SRR1273267	SRX534302	V3	NAFLD1	321, 039	310, 714
SRR1273268	SRX534303	V3	NAFLD2	221, 914	214, 505
SRR1273270	SRX534304	V3	NAFLD3	329, 871	319, 986
SRR1273272	SRX534305	V3	NAFLD4	173, 632	169, 602
SRR1273273	SRX534306	V3	NAFLD5	275, 969	269, 218
SRR1273275	SRX534307	V3	NAFLD6	554, 764	543, 413
SRR1273277	SRX534308	V3	NAFLD7	303, 189	294, 876
SRR1273279	SRX534309	V3	NAFLD8	160, 927	153, 241
SRR1273281	SRX534310	V3	NAFLD9	248, 792	246, 090
SRR1273282	SRX534311	V3	NAFLD10	170, 191	164, 909
Total	—	—	—	6, 621, 012	6, 440, 897


Inputs
Clean tags of Sample1, Sample2, …, SampleN
✓ Dereplication
Reference database, such as RDP
Sample size and abundance of unique tags
Sequences of multiple tags
Sequences of singleton tags
Classifier
Yes
Assigned tags
No
Unassigned tags
OTUs clustering guided by taxonomy
Append tags to the nearest OTUs
Pending tags
No
Yes
Chimera detection
Construction of OTUs
OTUs clustering by the improved heuristic search
Sample or group-based OTUs count table

Supplementary Figure S1: Pipeline of OTUs clustering by bioOTU.


Dataset of mock community (454)
Dataset of human stool community (Illumina)
✓Pre-processing by Mothur
Step 1: convert SRAs to SFF files
Step 2: quality filtering by *shhh.flows*
Step 3: length filtering by *trim.seqs*
✓Pre-processing by USEARCH
Step 1: merge paired reads by *-fastq_mergepairs*
Step 2: quality filtering by *-fastq_filter*
Clean tags for each sample and pooling
✓ bioOTU pipeline
Step 1: dereplication of clean tags
Step 2: taxonomy assignment of sequences
Step 3: taxonomy-guided OTUs clustering
Step 4: removals of chimeras
Step 5: heuristic OTUs clustering
Step 6: creation of OTUs table
✓Mothur pipeline
Step 1: dereplication by *unique.seqs*
Step 2: align sequences by *align.seqs*
Step 3: filter sequences by *filter.seqs*
Step 4: pre-clustering by *pre.cluster*
Step 5: detect chimera by *chimera.uchime*
Step 6: calculate distance by *dist.seqs*
Step 7: OTUs clustering by *cluster*
✓ UPARSE pipeline
Step 1: length trimming by *-fasta_trunclen*
Step 2: dereplication by *-derep_fulllength*
Step 3: pre-clustering by *–cluster_smallmem*
Step 4: remove singletons by *-sortbysize*
Step 5: OTUs clustering by *-cluster_otus*
Three profiles of the constructed OTUs

Supplementary Figure S2: Pre-processing steps and pipeline of OTUs clustering by bioOTU, Mothur, and UPARSE, respectively. The reads are retrieved from two communities (mock and real) and sequenced on different platforms (454 and Illumina). In UPARSE pipeline, the step of length trimming is not applied to Illumina reads.