SPARSE Documentation

Release 0.0.4

Zhemin Zhou, Nina Luhmann, Nabil-Fareed Alikhan, Mark Achtma

Jan 21, 2019

Getting Started:

SPARSE indexes >100,000 reference genomes in public databases into hierarchical clusters and uses it to predict origins of metagenomic reads.

CHAPTER 1

Installation guide

SPARSE runs on Unix and requires Python >= version 2.7

hardware

- SPASE runs best in multi-processes mode. Thus servers with at least 10-20 CPU cores were suggested.
- >= 300 GBytes of memory is required to handle over 10 million metagenomic reads.
- >= 500 GBytes of storage space.

System modules (Ubuntu 16.04):

- pip
- gfortran
- llvm
- libncurses5-dev
- cmake
- xvfb-run (for malt, optional)

3rd-party software:

- samtools (>=1.2)
- mash (>=1.1.1)
- bowtie2 (>=2.3.2)
- malt (>=0.4.0) (optional)

See requirements.txt for python module dependencies.

1.1 Installation with a Conda environment (Ubuntu)

To install SPARSE and all its dependencies in an isolated conda environment, you can use the environment file provided.

```
git clone git@github.com:zheminzhou/SPARSE.git
cd SPARSE/envs
conda env create -f sparse_env.yml
source activate sparse
```

1.2 Installation via PIP

```
pip install meta-sparse
```

1.3 Install from source files (Ubuntu)

```
sudo apt-get update
sudo apt-get install gfortran llvm libncurses5-dev cmake python-pip samtools bowtie2
git clone https://github.com/zheminzhou/SPARSE
cd SPARSE/EM && make
pip install -r requirements.txt
```

Change the parameters if needed.

1.4 Updating SPARSE

To update SPARSE, move to the installation directory and pull the latest version:

```
cd SPARSE
git pull
```

CHAPTER 2

Toy example of SPARSE

There is a bash script in the examples folder that runs through an example of SPARSE. This example takes reads from an ancient DNA sample of 800 year old *Salmonella enterica* from this publication.

The contents of the shell script download and create a Bowtie database of some *Salmonella* genomes (01-03) and then maps some of the ancient reads and displays a report.

```
sparse init --dbname toyset
sparse index --dbname toyset --seqlist Salmonella_toyset.txt
sparse query --dbname toyset --tag m==a | python ../SPARSE.py mapDB --dbname toyset --
↪mapDB Salmonella --seqlist stdin
sparse predict --dbname toyset --r1 Ragna.sample.fq.gz --workspace Ragna_toy --mapDB␣
↪Salmonella
sparse report Ragna_toy
cat Ragna_toy/profile.txt
sparse extract --workspace Ragna_toy --ref_id 10
```

CHAPTER 3

Parameter settings

All the default parameters are stored in parameter.py. Users do not need to specifiy these parameters in most of cases. Some parameters here may need to be specified during installation, while others can be specified for each database or for each SPARSE run.

3.1 Installation parameters

Parameters that need to be specified during installation, You need only point *BIN* to a folder that contains all the executables of the dependencies, e. g.

- *BIN = '/usr/local/bin/'*

Alternatively, if you have all executables in the system environmental parameter *$PATH*, use

- *BIN = ''*

You can also specify a pointer for each executable file :

- *mash = '{BIN}mash',*
- *bowtie2 = '{BIN}bowtie2',*
- *bowtie2_build = '{BIN}bowtie2-build',*
- *samtools = '{BIN}samtools',*
- *malt_run = 'xvfb-run –auto-servernum –server-num=1 {BIN}malt-run',*
- *malt_build = 'xvfb-run –auto-servernum –server-num=1 {BIN}malt-build',*

3.2 Runtime parameters

The following parameters that can be specified on-fly. You can also specify there default values for each database in: */path/to/sparse/database/dbsetting.cfg*

- *mismatch = 0.05* # mismatch parameter is used in the probalistic model. Given a higher value will report less bins
- *n_thread = 20* # number of threads for SPARSE. Higher value can accelerate the program
- *minFreq = 0.0001* # Minimum frequencies of a strain to be reported. Use minFreq = 0.000001 for ancient DNA samples
- *minNum = 10* # Minimum number of specific reads to report a strain. Use * minNum = 5 or less for ancient DNA samples
- *HGT_prior = [[0.05, 0.99, 0.1], [0.02, 0.99, 0.2], [0.01, 0.99, 0.5]]* # parameters to identify core genomic regions. Suggest to use default values
- *UCE_prior = [487, 2000]* # parameters to identify ultra-conserved elements. Suggest to use default values

3.3 Advanced parameters

Parameters to construct SPARSE databases, only for advanced uses:

- *msh_param = '-k 23 -s 4000 -S 42'* # change the parameter for the MASH program. reduce k and s accelerate the database indexing while bring in slightly more incorrect clusterings
- *# following three parameters are pointers to corresponding sub-folders. Change them if you want the actual data in a different folder than the database*
- *mash_db = '{dbname}/mash_db'*
- *bowtie_db = '{dbname}/bowtie_db'*
- *placer_db = '{dbname}/placer_db'*
- *taxonomy_db = '{dbname}/taxonomy'*

Parameters for hierarchical clustering levels:

- *barcode_dist = [0.1, 0.05,0.02,0.01, 0.005,0.002,0.001, 0.0005]*
- *barcode_tag = ['u', 's','r','p' , 'n','m','e' , 'c','a']*
- *representative_level = 2*

These parameters are for experts, and have not been tested for varied values

- *SPARSE = sparse_folder*
- *ipopt = '{SPARSE}/EM/solve-model'*
- *db_columns = ['index', 'deleted', 'barcode', 'sha256', 'size']*
- *metadata_columns = ['assembly_accession', 'version', 'refseq_category', 'assembly_level', 'taxid', 'organism_name', 'file_path', 'url_path']*
- *taxa_columns = ['subspecies', 'species', 'genus','family', 'order', 'class', 'phylum', 'kingdom', 'superkingdom'],*

CHAPTER 4

RefSeq database

The refseq database from NCBI stores >100,000 complete genomes and drafts that compass all tree of life. We firstly construct an empty database folder and assigns default control parameters for the database.

```
sparse init --dbname refseq
```

4.1 Index refseq database or update an exising database

A second command allows SPARSE to download all genomes in refseq on-fly and construct the database. The efficiency of the indexing process depends on both the downloading speed and the number of assigned CPUs. When assigning 20 CPUs, you can expect the whole process to finish in about one day.

```
sparse index --dbname refseq --update
```

Be aware that the newly added genomes are not ready for metagenomic reads. You need to run another command to update your representative databases.

We also release a pre-compiled database named "refseq_20180519", on the basis of NCBI RefSeq at 2018.05.19, at http://enterobase.warwick.ac.uk/sparse/

This database contains the MASH indexed master database and four default mapping databases:

```
representative
subpopulation
Virus
Eukaryota
```

As well as reference genomes for three important animal hosts:

```
Human
Swine
Bovine
```

To use the database, just download and untar the package (~350 GB):

```
curl -o refseq_20180519.tar.gz http://enterobase.warwick.ac.uk/sparse/refseq_20180519.
↪tar.gz
tar -vxzf refseq_20180519.tar.gz
```

4.2 Custom databases

You can also create a custom database, or add in custom genomes to an old database.

CHAPTER 5

Representatives database

In order to do read-level taxonomic binning, representative databases need to be compiled. Four default databases were designed cover most of the genetic diversities in metagenomic samples.

ANI 98% database for bacteria and archaea

```
sparse query --dbname refseq --default representative | python SPARSE.py mapDB --
↪dbname refseq --seqlist stdin --mapDB representative
```

ANI 99% database for bacteria and archaea (always use together with representative database)

```
sparse query --dbname refseq --default subpopulation | python SPARSE.py mapDB --
↪dbname refseq --seqlist stdin --mapDB subpopulation
```

ANI 99% virus database

```
sparse query --dbname refseq --default Virus | python SPARSE.py mapDB --dbname refseq␣
↪--seqlist stdin --mapDB Virus
```

ANI 99% eukaryota database (genome size <= 200MB)

```
sparse query --dbname refseq --default Eukaryota | python SPARSE.py mapDB --dbname␣
↪refseq --seqlist stdin --mapDB Eukaryota
```

Custom databases

In order to index a differet set of references into a representative database, see [here](custom.md)

CHAPTER 6

Building custom representative databases

You can also custom the representative databases. Here a human genome is used as an example:

We first query its record in a SPARSE refseq database using the assembly accession:

```
sparse query --dbname refseq_20171014 --assembly_accession GCF_000001405.37 > human.
↪tsv
```

The resulting file is:

```
index       deleted barcode sha256  size    assembly_accession      version refseq_
↪category assembly_level  taxid   organism_name   file_path       url_path        ␣
↪subspecies      species genus   family  order   class   phylum  kingdom superkingdom

107460      -       u107460.s107460.r107460.p107460.n107460.m107460.e107460.c107460.
↪a107460 d236b7835a3f10e596f9ce3c1f988b9e897f2dea216fd3dcde880eb91963863e        ␣
↪3253848404      GCF_000001405.37        37      reference genome        Chromosome ␣
↪    9606    Homo sapiens    -       ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/
↪001/405/GCF_000001405.37_GRCh38.p11/GCF_000001405.37_GRCh38.p11_genomic.fna.gz   - ␣
↪     Homo sapiens    Homo    Hominidae       Primates        Mammalia        ␣
↪Chordata        Metazoa Eukaryota
```

This file can be used as an input to build a new representative database named "Human":

```
sparse mapDB --dbname refseq --mapDB Human --seqlist human.tsv
```

Metagenomic reads are assigned using these representative databases, details see section on "read-level prediction".

CHAPTER 7

MASH based taxonomic assignment for genomic assemblies or read sets

SPARSE allows ultra-efficient taxonomic assignment with genomic assemblies or read sets, by using MASH to approximate average nucleotide identities (ANI).

genomic assembly (fasta format):

```
sparse mash --dbname refseq --query <assembly file>
```

Read set (in fastq format, either gzipped or not) :

```
sparse mash --dbname refseq --query <read file> --read
```

CHAPTER 8

Map metagenomic reads onto representative databases

Template of how to map metagenomic reads onto a representative database:

```
sparse predict --dbname </path/to/SPARSE/database> --mapDB <comma delimited MapDB's> -
↪-r1 <read_1> --r2 <read_2> --workspace <workspace_name>
```

Example (single end):

```
sparse predict --dbname refseq --mapDB representative,subpopulation,Virus --r1 read1.
↪fq.gz --workspace read1
```

The outputs consist of two files, with detailed information in the [“output” section](output.md).

8.1 Extract reference specific reads

You first need to find out the indices of the interesting references in the [output files](output.md), and use the indexes to extract related reads.

```
sparse extract --dbname refseq --workspace read1 --ref_id <comma delimited indices>
```

For example, we extract all reads specific to reference id 16, which is a Vibrio cholerae genome.

```
sparse extract --dbname refseq --workspace read1 --ref_id 16
```

CHAPTER 9

Outputs

9.1 Output for 'sparse predict'

The taxonomic profiling results for 'sparse query' are saved in <workspace>/profile.txt

The first three rows in 'profile.txt' summarize the status of the reads from the metagenomic sample:

```
Total <No. reads>     <No. matched reads>
Unmatched     <% unmatched reads in total reads>      0.000
Uncertain_match       <% unreliable matches in total reads>   <% unreliable matches␣
↪in total matches>
```

Example:

```
Total   26726530   23783388.0
Unmatched   11.012   0.000
Uncertain_match   36.102   40.570
```

This example suggests that 89% of reads are matched against at least one reference in the database. Subsequently, 60% of all matches found are used for taxonomic predictions.

The following lines describe the prediction at different taxonomic levels, in the following format:

```
<SPARSE group>   <% in total reads>   <% in matched reads>   <taxonomic labels>   (
↪<reference IDs>)
```

Example:

```
~154    2.1706  2.4392  Bacteria|-
↪|Actinobacteria|Actinobacteria|Micrococcales|Micrococcaceae (15969,66991,66935,
↪66915,67189,110179,40981,154,67166,67220,114405,66878,66930,82153,40861,40710,67029)
u154    2.1701  2.4387  Bacteria|-
↪|Actinobacteria|Actinobacteria|Micrococcales|Micrococcaceae|Rothia (15969,66991,
↪66935,66915,67189,110179,40981,154,67166,67220,114405,66878,66930,82153,40861,40710,
↪67029)
```

(continues on next page)

(continued from previous page)

```
s154    2.1551  2.4217  Bacteria|-
↪|Actinobacteria|Actinobacteria|Micrococcales|Micrococcaceae|Rothia|Rothia␣
↪dentocariosa (*Rothia sp. HMSC067H10/*Rothia sp. HMSC064D08/*Rothia sp. HMSC071F11/
↪*Rothia sp. HMSC069C01) (15969,66991,66935,66915,67189,110179,40981,154,67166,67220,
↪114405,66878,66930,82153,40861,40710,67029)
~613    1.4988  1.6843  Bacteria|-
↪|Firmicutes|Negativicutes|Veillonellales|Veillonellaceae (16778,16416,117596,16415,
↪10931,17276,113949,60730,613)
u613    1.4934  1.6782  Bacteria|-
↪|Firmicutes|Negativicutes|Veillonellales|Veillonellaceae|Veillonella (16778,16416,
↪117596,16415,10931,17276,113949,60730,613)
s613    1.4507  1.6302  Bacteria|-
↪|Firmicutes|Negativicutes|Veillonellales|Veillonellaceae|Veillonella|Veillonella␣
↪parvula (*Veillonella sp. 6_1_27/*Veillonella sp. S13054-11/*Veillonella sp. 3_1_
↪44) (16778,16416,117596,16415,10931,17276,113949,60730,613)
r15969  0.4677  0.5256  Bacteria|-
↪|Actinobacteria|Actinobacteria|Micrococcales|Micrococcaceae|Rothia|Rothia␣
↪dentocariosa|- (15969)
p15969  0.3907  0.4391  Bacteria|-
↪|Actinobacteria|Actinobacteria|Micrococcales|Micrococcaceae|Rothia|Rothia␣
↪dentocariosa|-|Rothia dentocariosa M567: GCF_000143585.1 (15969)
r16416  0.1838  0.2065  Bacteria|-
↪|Firmicutes|Negativicutes|Veillonellales|Veillonellaceae|Veillonella|*Veillonella␣
↪sp. 6_1_27|- (16416)
p16416  0.1631  0.1833  Bacteria|-
↪|Firmicutes|Negativicutes|Veillonellales|Veillonellaceae|Veillonella|*Veillonella␣
↪sp. 6_1_27|-|Veillonella sp. 6_1_27: GCF_000163735.1 (16416)
```

The SPARSE groups are internal hierarchical clustering results stored in the SPARSE database. The group label consists of two components. The prefix presents the ANI level of the cluster and the following number presents the designation of the cluster.

For example, 's613' is a cluster '613' in 's' level (ANI 95%, "species level") The correlation between prefix and ANI level is:

```
~     <90% ANI
u     90% ANI
s     95% ANI
r     98% ANI
p     99% ANI
n     99.5% ANI
m     99.8% ANI
e     99.9% ANI
c     99.95% ANI
a     100% ANI
```

's' (ANI 95%) is normally treated as a 'gold standard' criterion for species definition.

For each SPARSE goup, the traditional taxonomic labels follow the format:

```
<superkingdom>|<kingdom>|<phylum>|<class>|<order>|<family>|<genus>|<species>|
↪<subspecies>|<reference_genome>
```

These taxonomic labels are summarised from the input database. Sometimes multiple species will be associated with one SPARSE group:

```
s613    1.4507  1.6302  Bacteria|-
↪|Firmicutes|Negativicutes|Veillonellales|Veillonellaceae|Veillonella|Veillonella␣
↪parvula (*Veillonella sp. 6_1_27/*Veillonella sp. S13054-11/*Veillonella sp. 3_1_
↪44) (16778,16416,117596,16415,10931,17276,113949,60730,613)
```

In this example, group s613 is associated with four different species:

```
Veillonella parvula
*Veillonella sp. 6_1_27
*Veillonella sp. S13054-11
*Veillonella sp. 3_1_44
```

Informal names are marked with prefix "*". The most probable species is shown first, and followed by the other three names in a bracket. There is another bracket after the taxonomic labels:

```
(16778,16416,117596,16415,10931,17276,113949,60730,613)
```

These are the IDs of the actual reference genomes that were found in the database. They can be used to extract reference specific reads using the command 'sparse extract'.

9.2 Output for 'sparse report'

sparse can provide a report that combines multiple 'sparse predict' runs together into a tab-delimited text file. This command also identifies potential pathogens in the predictions.

```
#Group  #Pathogenic     ERR1659111      ERR1659110      #Species        #Taxon
s3080   non     4.47309775569   4.84028327303   Actinomyces dentalis (*Actinomyces sp.
↪ oral taxon 414)  Bacteria|-
↪|Actinobacteria|Actinobacteria|Actinomycetales|Actinomycetaceae|Actinomyces|Actinomyces␣
↪dentalis (*Actinomyces sp. oral taxon 414)
s1438   non     0.821962806352  3.57658189557   Desulfomicrobium orale  Bacteria|-
↪|Proteobacteria|Deltaproteobacteria|Desulfovibrionales|Desulfomicrobiaceae|Desulfomicrobium|Desulfo
↪orale
s9975   non     2.04489272864   1.85184148971   *Anaerolineaceae bacterium oral taxon␣
↪439       Bacteria|-|Chloroflexi|Anaerolineae|Anaerolineales|Anaerolineaceae|-
↪|*Anaerolineaceae bacterium oral taxon 439
s939    non     1.81538010098   0.712860400235  Pseudopropionibacterium propionicum  ␣
↪  Bacteria|-
↪|Actinobacteria|Actinobacteria|Propionibacteriales|Propionibacteriaceae|Pseudopropionibacterium|Pse
↪propionicum
s8820   non     1.67063037869   0.491279312566  *Ottowia sp. Marseille-P4747␣
↪(*Ottowia sp. oral taxon 894)      Bacteria|-
↪|Proteobacteria|Betaproteobacteria|Burkholderiales|Comamonadaceae|Ottowia|*Ottowia␣
↪sp. Marseille-P4747 (*Ottowia sp. oral taxon 894)
s2215   non     1.31802856115   0.34575838713   Lautropia mirabilis     Bacteria|-
↪|Proteobacteria|Betaproteobacteria|Burkholderiales|Burkholderiaceae|Lautropia|Lautropia␣
↪mirabilis
s2590   non     0.665641018802  0.612783437737  Actinomyces cardiffensis        ␣
↪Bacteria|-
↪|Actinobacteria|Actinobacteria|Actinomycetales|Actinomycetaceae|Actinomyces|Actinomyces␣
↪cardiffensis
s2189   non     0.87220732902   0.296597041195  Corynebacterium matruchotii     ␣
↪Bacteria|-
↪|Actinobacteria|Actinobacteria|Corynebacteriales|Corynebacteriaceae|Corynebacterium|Corynebacteriu
↪matruchotii
```

(continues on next page)

(continued from previous page)

```
s108979 non     0.295928369726  0.857545958706  *Actinomyces sp. oral taxon 897␣
↪Bacteria|-
↪|Actinobacteria|Actinobacteria|Actinomycetales|Actinomycetaceae|Actinomyces|*Actinomyces␣
↪sp. oral taxon 897
```

The first line shows the samples in the report, as well as additional annotations (starts with '#'). #Group and #Taxon are identical to the 'sparse predict' output. #Species is a simple extraction of the most probably species in the #Taxon column and #Pathogenic contains potential pathogen predictions encoded as:

```
non  - not a pathogen
*    - commensal and normally not a pathogen
**   - Possibly a pathogen
***  - Pathogen
**** - Important pathogen, possibly fatal
```

The numbers shows the abundances of the species in each metagenomic read set. It is normally shown in percentages, unless parameter '–absolute' is applied, which changes the numbers to be absolute read counts.

The last row of the output is a summary of all unknown/uncertain reads without taxonomic classifications.

CHAPTER 10

Citation

SPARSE has been pulished as a conference paper in RECOMB 2018. Zhou Z., Luhmann N., Alikhan NF., Quince C., Achtman M. (2018) Accurate Reconstruction of Microbial Strains from Metagenomic Sequencing Using Representative Reference Genomes. In: Raphael B. (eds) Research in Computational Molecular Biology. RECOMB 2018. Lecture Notes in Computer Science, vol 10812. Springer, Cham. DOI: https://doi.org/10.1007/978-3-319-89929-9_15

A preprint can also be found in BioRxiv : https://www.biorxiv.org/content/early/2017/11/07/215707

CHAPTER 11

Indices and tables

- genindex
- modindex
- search