COMMET: comparing and combining multiple metagenomic datasets

Nicolas Maillet, Guillaume Collet, Thomas Vannier, Dominique Lavenier, Pierre Peterlongo

COMMET: comparing and combining multiple metagenomic datasets

Nicolas Maillet*, Guillaume Collet†, Thomas Vannier‡, Dominique Lavenier* and Pierre Peterlongo*⁕
*INRIA / IRISA-UMR CNRS 6074, EPI GenScale, Rennes, France
†INRIA / IRISA-UMR CNRS 6074, EPI Dyliss, Rennes, France
‡CEA Genoscope / CNRS UMR 8030 / Université d'Évry, Evry, France
⁕ Corresponding author: Pierre Peterlongo pierre.peterlongo@inria.fr

***Abstract*—Metagenomics offers a way to analyze biotopes at the genomic level and to reach functional and taxonomical conclusions. The bio-analyzes of large metagenomic projects face critical limitations: complex metagenomes cannot be assembled and the taxonomical or functional annotations are much smaller than the real biological diversity. This motivated the development of *de novo* metagenomic read comparison approaches to extract information contained in metagenomic datasets.**

However, these new approaches do not scale up large metagenomic projects, or generate an important number of large intermediate and result files. We introduce COMMET ("COmpare Multiple METagenomes"), a method that provides similarity overview between all datasets of large metagenomic projects.

Directly from non-assembled reads, all against all comparisons are performed through an efficient indexing strategy. Then, results are stored as bit vectors, a compressed representation of read files, that can be used to further combine read subsets by common logical operations. Finally, COMMET computes a clusterization of metagenomic datasets, which is visualized by dendrogram and heatmaps.

Availability: http://github.com/pierrepeterlongo/commet

I. INTRODUCTION

NGS revolution enabled the emergence of the metagenomic field where an environment is sequenced instead of an individual or a species, opening the way to a comprehensive understanding of environmental microbial communities. Large metagenomic projects such as MetaSoil [1], MetaHit [2] or Tara Oceans [3] witness this evolution. Analyzing of metagenomic data is a major bottleneck. For instance, assembly tests over "simple" simulated metagenomes showed that N50 is only slightly larger than read sizes [4]. This situation becomes even worth on complex datasets, such as seawater, where millions of distinct species coexist. In this case, biodiversity can be estimated by using statistical approaches [5] or by mapping reads on reference banks [6], [7]. Nevertheless, statistical approaches are limited to a few dozens of species with limited differences in their relative abundance. In addition, the mapping approaches are limited to current knowledge contained in reference banks that suffer from their incompleteness and their inherent errors [8].

A key point of substantial metagenomic projects stands in the number of metagenomes they produce. Then, similarities and differences between metagenomes can be exploited as a source of information, measuring external effects like pollution sources, geographic locations, and patient microbial gut environment [2], [9]. A few methods were proposed to compare metagenomes using external information sources such as taxonomic diversity [10] or functional content [11]. However, these methods are biased because as they are based on partial knowledge.

Methods were proposed to compare metagenomes without using any *a priori* knowledge. These *de novo* methods use global features like GC content [12], genome size [13] or sequence signatures [14]. Theses methods face limitations as they are based on rough imprecise criteria and as they only compute a similarity distance: they do not extract similar elements between samples. We believe that it is possible to go further by comparing metagenomic samples at the read sequence level. This provides a higher precision distance and, importantly, it provides reads that are similar between datasets or that are specific to a unique dataset, enabling their latter analysis: assembly with better coverage or comparison with other metagenomic samples. Such comparisons may be performed using Blast [15] or Blat [16] like tools. Unfortunately, these methods do not scale up on large comparative metagenomic studies in which hundreds of millions of reads have to be compared to other hundreds of millions of reads. For instance, one can estimate that comparing a hundred of metagenomes each composed by a hundred of millions of reads of size 100 would require centuries of CPU computation. The crAss approach [17] constructs a reference metagenome by cross assembling reads of all samples. Then, it maps the initial reads on the so obtained contigs and several measures are derived, based on the repartition of mapped reads. This method provides results of high quality. However, due to its assembly and mapping approach, it does not scale up to large metagenomic datasets. Simpler methods such as TriageTools [18] or Compareads [19] measure the sequence similarity of a read with a databank by counting the number of k-mers (words of length k) shared with the databank. Due to memory consumption, TriageTools cannot use k values larger than 15 and is thus limited to small datasets (a few hundred of thousands reads of length 100). The Compareads tool scales up to large datasets with a small memory footprint and acceptable running time. However, applied on large metagenomic projects, this tools generates an important number of large intermediate result files. In practice, applying Compareads to N datasets generates N^2 resulting new datasets, each of the size of the original ones at worse. Additionally, Compareads leads to highly redundant computation raising up the execution time. These drawbacks are serious bottlenecks limiting the practical usage of Compareads.

In this paper, we introduce COMMET ("COmpare Multiple METagenomes"), a fast software that provides a global similarity overview between all datasets of a metagenomic project. COMMET is based on the Compareads philosophy that consists in determining similarity between two metagenomic datasets by extracting common reads using k-mer approach: two reads are considered similar if they share t non-overlapping k-mers (t and k are parameters). A metagenomic project involving N datasets will thus requires the computation of N^2 intersections which is both time- and storage-consuming. To keep computation time as low as possible, the computation of the N^2 intersections has been strongly improved compared to the Compareads approach through an efficient indexing strategy in which each file is fully indexed only once. In addition, to save storage space, intersections between metagenomic datasets are represented as bit vectors. This compact representation reduces the storage space by two orders of magnitude. Moreover, it provides an easy way to filter and sub-sample reads, or to combine various results by applying logical operations. Finally, COMMET computes a clusterization of metagenomic datasets, which is visualized by dendrogram and heatmaps.

II. METHOD

A. Comparing two sets of reads

The COMMET algorithm to compare two sets of read is based on the Compareads [19] methodology. It consists in finding reads from a set A that are similar to at least one read from a set B. The similarity between two reads is based on a minimal number t of non-overlapping identical k-mers. This core operation is directed : it provides reads from A similar to reads from B but it does not provides reads from B similar to reads from A. Note that, as explained below, this operation is based on a heuristic. Thus we denote this operation by $A \overrightarrow{\cap} B$.

Computing $A \overrightarrow{\cap} B$ consists in two steps. Firstly, k-mers from B are indexed in a Bloom filter like data-structure [20]. Secondly, non-overlapping k-mers of reads from A are searched in the Bloom filter. A read r from A sharing t non-overlapping k-mers with the Bloom filter is considered similar to at least one read from B. However, the algorithm does not check that these k-mers co-occur on a single read from B, which is a source of false positives. Readers are invited to refer to [19] for having more details on precision results.

We recall that the following strategy is applied in order to limit the second source of false positives. First $A \overrightarrow{\cap} B$ is computed. Then, instead of navely computing $B \overrightarrow{\cap} A$, $B \overrightarrow{\cap} (A \overrightarrow{\cap} B)$ is computed. This limits the indexed reads of A to those already detected as similar to at least one read from B. Finally, the symmetrical operation is performed: $A \overrightarrow{\cap} (B \overrightarrow{\cap} (A \overrightarrow{\cap} B))$.

The previously exposed strategy to fully compare sets A and B within three consecutive $\overrightarrow{\cap}$ operations has also the advantage to limit the indexation effort. Indeed, only the first $A \overrightarrow{\cap} B$ operation indexes the full set B. The two other operations only index subsets of A and B.

While comparing read samples A and B, the final results of interest are the reads of A similar to reads of B computed by $A \overrightarrow{\cap} (B \overrightarrow{\cap} (A \overrightarrow{\cap} B))$ and reads of B similar to reads of A computed by $B \overrightarrow{\cap} (A \overrightarrow{\cap} B)$. For sake of simplicity, we denotes these two sets as, respectively, $A \overrightarrow{\Cap} B$ and $B \overrightarrow{\Cap} A$.

In the following sections we present the COMMET novelties: represent read subsets with a limited disk space impact, new read filtering and read subsets manipulation features, compare multiple sets of reads, visualize dataset's similarities as heatmaps and dendrogram.

B. Read subsets representation

In COMMET we propose a simple yet compact data structure to represent a read subset: a vector of bits where each bit represents a read of the original read set. This is what we call the "bit vector representation". As shown below, this representation enables to filter and to subsample read files, to represent $\overrightarrow{\cap}$ (and thus $\overrightarrow{\Cap}$) results and to easily perform logical operation between read subsets.

Note that with such a representation, a bit vector needs hundreds to thousand times less disk space than a classical uncompressed fastq file. Note also that this way of coding read subsets is not limited to the COMMET framework. It may be applied to any other programs that manipulate read subsets. Thus, the COMMET tool includes a C++ library of reusable components to manipulate read subsets.

In the COMMET framework, the bit vector representation is used as inputs and/or outputs of all tools. In particular they are used in the following operations:

1) Read subsampling and filtering: With huge datasets, it may appear necessary to subsample, for instance limiting each read file to a same number m of a few millions reads. This is immediate by creating a bit vector in which only the first m bits are set to 1, while others are set to 0.

Raw NGS reads also usually need to be filtered on several practical characteristics (read size, read complexity, ...). Thus, a bit vector is a direct representation of a filtered result: bit values associated to selected reads are set to 1, the others to 0. A combination of subsampling and filtering allows to select only the m first reads that fulfill the filtration criteria.

2) Representing the similar reads: Results of any $\overrightarrow{\cap}$ operation is represented by a bit vector. Bit values of reads from the query set detected as similar to at least one read from the reference set are set to 1 and the others are set to 0.

3) Compute logical operations on read subsets: The bit vector representation is ideally suited to perform fundamental logical operations. COMMET provides a module to perform the AND, OR and NOT operations between distinct subsets of a single initial set of reads.

As presented in the simple case study (Section II-F), these operations, although simple, are powerful while dealing with read subsets. They allow to combine comparison results and so to focus on read subsets intersections or exclusions.

These logical operations perform very efficiently, both in terms of execution time and memory footprint. Moreover, it is worth to notice that they do not generate large result files, as results of these logical operations are also represented as bit vectors. This allows to intensively manipulate read subsets with no technical limitations.

C. Dealing with more than two datasets

We recall that the computation of the $A \overrightarrow{\cap} B$ core operation involves indexation and search. Once the k-mers of the reads from B are indexed, then the k-mers of the reads from A are sequentially search in the index. If more than a threshold number t of such k-mers are find in the index, then the given read from A is considered as similar to a read from B, which means that the associated value in the bit vector is set to 1.

Consider $S = \{R_1, \ldots, R_N\}$ a set of $N \geq 2$ read sets. Applying COMMET on the whole S implies that $\forall (i,j) \in [1,N]^2, i < j$, three ordered operations are performed:

1) $R_i \overrightarrow{\cap} R_j$
2) $R_j \overrightarrow{\Cap} R_i = R_j \overrightarrow{\cap} (R_i \overrightarrow{\cap} R_j)$
3) $R_i \overrightarrow{\Cap} R_j = R_i \overrightarrow{\cap} (R_j \overrightarrow{\cap} (R_i \overrightarrow{\cap} R_j))$

Note that for each couple (i,j), the order (i,j) or (j,i) only slightly changes the overall results of the three operations. To avoid redundancies, we limit these operations to $i < j$.

1) Factorizing the indexation: In practice, applying COMMET on S implies to perform the $R_i \overrightarrow{\cap} R_j$ operations for all $i < j$. In particular, $R_1 \overrightarrow{\cap} R_N \ldots R_{N-1} \overrightarrow{\cap} R_N$ have to be computed. For these $N-1$ computations, the k-mer index of R_N is the same. To avoid redundancies, the R_N index is computed only once and the $N-1$ remaining sets are compared to R_N using this single index. In general, while R_{ref} ($ref \in [2,N]$) is indexed, the index is conserved in RAM memory during the computation of the $ref - 1$ comparisons $R_{query} \overrightarrow{\cap} R_{ref}$, with $query < ref$.

2) Results visualization: Comparisons of $N \geq 2$ read sets $\{R_1, \ldots, R_N\}$ provide useful metrics that give an overview of the genomic diversity of the studied samples. Those metrics are summarized in three matrices M_1, M_2, and M_3 with values calculated as follows:

- $M_1(i,j) = |R_i \overrightarrow{\Cap} R_j|$
- $M_2(i,j) = 100 \times \frac{|R_i \overrightarrow{\Cap} R_j|}{|R_i|}$
- $M_3(i,j) = 100 \times \frac{|R_i \overrightarrow{\Cap} R_j| + |R_j \overrightarrow{\Cap} R_i|}{|R_i| + |R_j|}$.

$M_1(i,j)$ with $(i,j) \in [1,N]^2$, is the raw number of reads from R_i that are similar to at least one read from R_j. As read sets may be of different sizes, $M_2(i,j)$ is the percentage of reads from R_i similar to at least one read from R_j. Those two first matrices are asymmetrical. M_3 is a symmetrical matrix. $M_3(i,j)$ is the percentage of similar reads between the two sets with respect to the total number of reads in R_i and R_j.

For each matrix, a heatmap is generated . Additionally, M_3 is used to construct a dendrogram representation by hierarchical clustering (see Fig 2 for an example of a heatmap and a dendrogram generated by COMMET).

D. The COMMET modules

COMMET integrates four independent modules written in C++, all manipulating, as inputs and outputs, the bit vector representation of read subsets. Additionally, COMMET provides a python script (Commet.py) that takes $N \geq 2$ read sets, filters them, compares them and generates explicit representations of comparative results, see Section II-E.

1) Filtering and subsampling reads: Thanks to the first module, $filter\_reads$, each read of each dataset (fasta or fastq format, gzipped or not) is filtered out according to user-defined criteria: minimal read length, number of undefined bases, and Shannon complexity [21], used to remove low complexity sequences. The result is a bit vector for each input read file. $Filter\_reads$ can also subsample each read set by limiting the number of selected reads to a user defined parameter m. The m first reads that passed the filters are selected.

2) Performing the $\overrightarrow{\cap}$ core operation: The second module, $index\_and\_search$, performs the $\overrightarrow{\cap}$ core operation, representing results using the bit vector representation. It inputs a set of read sets (the queries) to be searched in an indexed read set (the bank). A read set may be composed of several read files. Each file could be associated to a bit vector. In this latter case, $index\_and\_search$ only considers reads whose associated bit values are set to 1.

3) Manipulating read subsets: The third module, $bvop$ (bit vector operations) inputs one or two bit vectors. In this second case, the two bit vectors should represent subsets of the same initial set. This module performs the NOT operation on a single bit vector, and the AND, OR, and $AND\ NOT$ operations on two bit vectors.

4) From bit vectors to read files: Given an original read file and its bit vector, the last module, $extract\_reads$, generates an explicit representation of any read subset.

E. Automatization for $N \geq 2$ read sets

COMMET includes a python script (Commet.py) which inputs $N \geq 2$ read sets. This pipeline i) filters reads, given user-defined parameters, ii) compares all-against-all read sets, and iii) outputs a user-friendly visualization of results. The outputs consist in the three matrices in csv format, their heatmaps and a dendrogram as described in Section II-C2. The dendrogram is realized using the *hclust R* function, computing a hierarchical complete clustering.

F. Combining read subsets use case

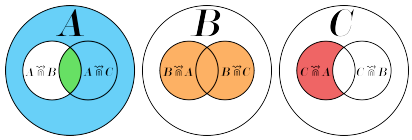

Fig. 1. Logical operations on intersections between A, B and C extract read subsets of interest. The blue subset corresponds to the $A\ AND\ NOT(A \overrightarrow{\Cap} B)$ operation. The green subset corresponds to the $(A \overrightarrow{\Cap} B)\ AND\ (A \overrightarrow{\Cap} C)$ operation. The orange subset corresponds to the $(B \overrightarrow{\Cap} A)\ OR\ (B \overrightarrow{\Cap} C)$ operation. The red subset corresponds to the $(C \overrightarrow{\Cap} A)\ AND\ NOT(C \overrightarrow{\Cap} B)$ operation.

By using the $bvop$ module, logical operations can be performed between inputs/outputs of the COMMET pipeline output. For instance, reads from A not similar to any read from set B (blue subset of Fig 1) are obtained by first applying $NOT(A \overrightarrow{\Cap} B)$ operation. Reads from A similar to at least one read from B and one read from C (green subset

TABLE I. 28 METAGENOMES FROM THE IMG/M DATABASE

Identifiers	Description
SWITGRA	Rhizosphere soil from *Panicum virgatum*
SUBGIN	Oral TM7 microbial community of Human
TERMITE2, TERMITE1	Gut microbiome of divers termites
SOILM, SOILD, SOILL	Soil microbiome from divers locations
OMIN, MESO, EUPHO	Divers marine planktonic communities
BEETLE	*Dendroctonus ponderosae*
ACOFUNT, ACOFUNB, CLOFUN, ACEFUN, TRAFUN, FUNCOMB	Fungus garden of divers ants
FUNTER	Fungus-growing termite worker
WALLABY	Forestomach microbiome of tammar wallaby
RICE	Endophytic microbiome from rice
SNAIL	*Achatina fulica*
SNOCT	*Sirex noctilio* microbiome
XALARV, XAAD	*Xyleborus affinis* microbiome (larvae, adult)
HGUT7, HGUT8	Human gut community
PANDA2, PANDA5	Wild panda gut microbiome

of Fig 1), are identified by computing the AND operation: $(A \overset{\rightarrow}{\widehat{\cap}} B)\ AND\ (A \overset{\rightarrow}{\widehat{\cap}} C)$. In the same spirit, reads from B similar to at least one read from A or one read from C (orange subset of Fig 1), are found by computing the OR operation: $(B \overset{\rightarrow}{\widehat{\cap}} A)\ OR\ (B \overset{\rightarrow}{\widehat{\cap}} C)$. Operations may be combined to obtain more complex results as, for instance, the red subset of Fig 1, representing reads from C similar to at least one read from A, but not similar to any read from B. This would be done by applying the $(C \overset{\rightarrow}{\widehat{\cap}} A)\ AND\ NOT (C \overset{\rightarrow}{\widehat{\cap}} B)$ operation.

III. RESULTS

A. COMMET *efficiently compares multiple metagenomes*

We tested COMMET on a set of 28 metagenomes from the IMG/M database [7] (see Table I). These 28 metagenomes were compared with options $k = 33$, $t = 2$ and $m = 10000$. Computations were done using COMMET (Commet.py) and Compareads (v1.3.1) on a 2.9 GHz Intel Core i7 processor with 8GB of RAM and a Solid-State Drive. COMMET calculated the 756 intersections in 35 minutes while Compareads took 81 minutes. In this experiment, COMMET is 2x faster than Compareads thanks to its indexing strategy (each file is fully indexed only once). The obtained dendrograms, shown in Figure 2, are biologically coherent. The different fungus samples are grouped together as well as soil samples, marine planktonic communities and insects. The two human gut microbiome samples are far from other species, as well as the two panda gut microbiome samples.

B. Metasoil study

The MetaSoil study focuses on untreated soils of Park Grass Experiment, Rothamsted Research, Hertfordshire, UK. One of the goals of this study is to assess the influence of depth, seasons and extraction procedure on the sequencing [22]. To achieve this, the 13 metagenomes from MetaSoil, two other soil metagenomes and a sea water metagenome, were compared at the functional level using MG-RAST [23]. This approach identified 835 functional subsystems present in at least one of those metagenomes. On Figure 3.a, samples were clustered using the relative number of reads associated with the 835 functions. This figure shows that the extraction procedure correlates with sample clusters: two metagenomic samples processed with the same extraction procedure share more similarities at the functional level than two samples processed with different extraction procedures [1].

This study was reproduced with COMMET on all available metagenomes. The generated bit vectors weigh 68MB while

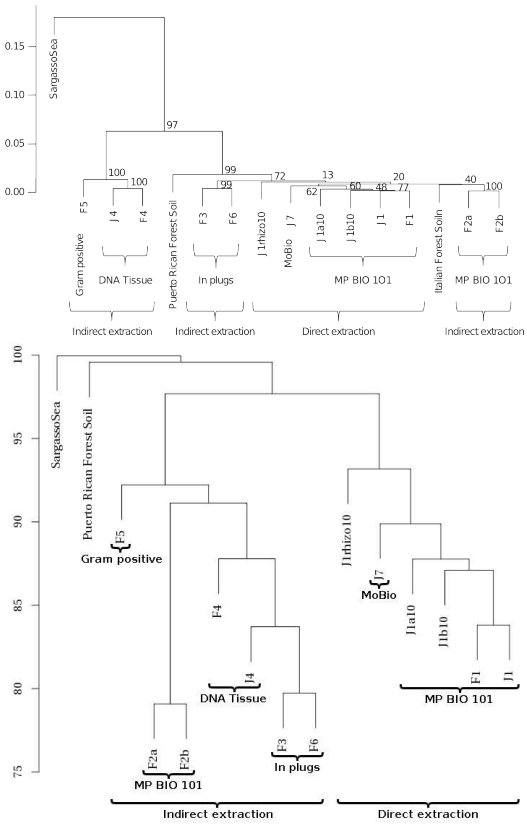

Fig. 3. Dendrograms from MetaSoil study (top, figure from [1]) and COMMET analysis (bottom), comparing the 13 MetaSoil samples, an other soil metagenome and a seawater metagenome (Sargasso Sea).

the explicit representation of the fasta results requires 6.4GB. The storage footprint is thus divided by a factor 100. This ratio is even higher if using fastq format or if dealing with larger read files. The COMMET computation time was 828 minutes (the same set treated by Compareads took 2981 minutes).

Although COMMET uses another metric, the produced dendrogram is highly similar to the MetaSoil one (see Fig 3). On both dendrograms, samples coming from direct extraction are clustered together and external metagenomes are far from the MetaSoil's. Moreover, on the COMMET dendrogram, all samples coming from indirect extraction are clustered together, which is not the case in the MetaSoil study. Even if the two comparing methods are different, they lead to the same conclusion: extraction procedures have a critical impact on sequencing.

IV. CONCLUSION

COMMET gives a global similarity overview of all datasets of a large metagenomic project. It performs all-against-all comparisons of N datasets by factorizing indexation phases. Disk I/Os and storage footprint are highly limited thanks to a new read subset representation which reduces the storage space by at least two orders of magnitude compare to explicit fasta or fastq format. Interestingly, this read subset representation is a powerful way to compute extremely fast boolean operations between read subsets without copying large read files. This enables to focus on reads that fulfill several distinct constraints of interest. The advantages of this representation and of

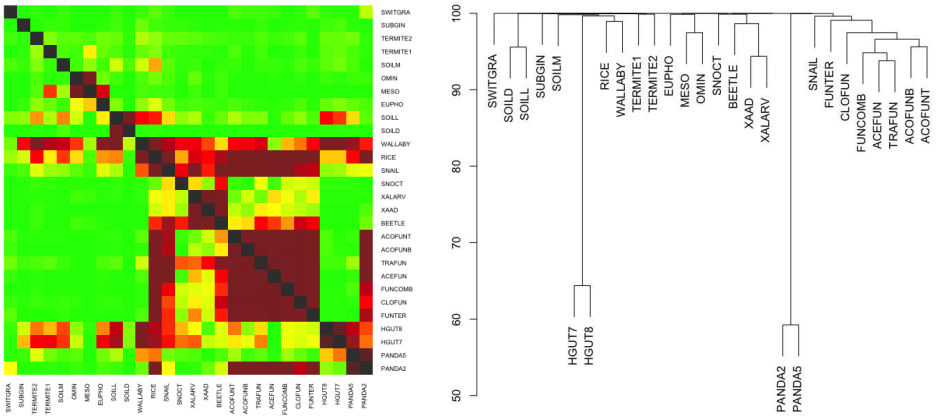

Fig. 2. Heatmap (left) and dendrogram (right) representation of the results of the comparison of 28 datasets from the IMG/M database. Results are given with $t = 2$, $m = 10000$ and $k = 33$. The heatmap is constructed from the matrix M_2 and is thus asymmetrical. The dendrogram is constructed from the matrix M_3 by the hierarchical clustering procedure available in *R* (method "complete").

the boolean manipulation are not limited to the COMMET framework. Thus, COMMET includes a C++ library of reusable components to manipulate read subsets.

COMMET produces graphical outputs that sum up all-against-all comparisons results and open the way for further statistical analysis, thanks to the provided similarity matrices.

A future work will consists in quickly identify significant clusters of read sets by applying rougher comparative metrics (such as the GC content) or a statistical framework based on Principal Component Analysis (PCA). Then, COMMET should be used to go further by precisely compute the shared reads between read sets inside clusters, or between clusters.

COMMET is available under the A-GPL license: http://github.com/pierrepeterlongo/commet.

ACKNOWLEDGMENT

Authors warmly thank Claire Lemaitre for her precious advices and her help designing the *R* functions. This work was supported by the french ANR-2010-COSI-004 *MAPPI* and by the ANR-12-BS02-0008 *Colibread* projects. Guillaume Collet's work is funded by the investment expenditure program IDEALG 1192 ANR-10-BTBR-02-04-11.

REFERENCES

[1] T. O. Delmont *et al.*, "Structure, fluctuation and magnitude of a natural grassland soil metagenome," pp. 1677–1687, 2012.

[2] J. Qin *et al.*, "A human gut microbial gene catalogue established by metagenomic sequencing," *Nature*, vol. 464, no. 7285, pp. 59–65, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1038/nature08821

[3] E. Karsenti *et al.*, "A holistic approach to marine Eco-systems biology," *PLoS Biology*, vol. 9, 2011.

[4] M. Pignatelli and A. Moya, "Evaluating the fidelity of De Novo short read metagenomic assembly using simulated data," *PLoS ONE*, vol. 6, 2011.

[5] Y. Wang *et al.*, "Metacluster 5.0: A two-round binning approach for metagenomic data for low-abundance species in a noisy sample," *Bioinformatics*, vol. 28, 2012.

[6] D. H. Huson *et al.*, "MEGAN analysis of metagenomic data." *Genome research*, vol. 17, pp. 377–386, 2007.

[7] V. M. Markowitz *et al.*, "IMG/M: The integrated metagenome data management and comparative analysis system," *Nucleic Acids Research*, vol. 40, 2012.

[8] A. M. Schnoes *et al.*, "Annotation error in public databases: Misannotation of molecular function in enzyme superfamilies," *PLoS Comput Biol*, vol. 5, no. 12, p. e1000605, Dec 2009.

[9] T. O. Delmont, P. Simonet, and T. M. Vogel, "Describing microbial communities and performing global comparisons in the omic era," pp. 1625–1628, 2012.

[10] S. Jaenicke *et al.*, "Comparative and joint analysis of two metagenomic datasets from a biogas fermenter obtained by 454-pyrosequencing," *PLoS ONE*, vol. 6, 2011.

[11] M. O. A. Sommer, G. Dantas, and G. M. Church, "Functional characterization of the antibiotic resistance reservoir in the human microflora." *Science (New York, N.Y.)*, vol. 325, pp. 1128–1131, 2009.

[12] K. U. Foerstner *et al.*, "Environments shape the nucleotide composition of genomes." *EMBO reports*, vol. 6, pp. 1208–1213, 2005.

[13] J. Raes *et al.*, "Prediction of effective genome size in metagenomic samples." *Genome biology*, vol. 8, p. R10, 2007.

[14] B. Jiang *et al.*, "Comparison of metagenomic samples using sequence signatures." *BMC genomics*, vol. 13, p. 730, Jan. 2012.

[15] S. F. Altschul *et al.*, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022283605803602

[16] W. J. Kent, "BLAT—The BLAST-Like Alignment Tool," *Genome Research*, vol. 12, no. 4, pp. 656–664, Mar. 2002. [Online]. Available: http://www.genome.org/cgi/doi/10.1101/gr.229202

[17] B. E. Dutilh *et al.*, "Reference-independent comparative metagenomics using cross-assembly: crAss." *Bioinformatics (Oxford, England)*, vol. 28, no. 24, pp. 3225–31, Dec. 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/23074261

[18] D. Fimereli, V. Detours, and T. Konopka, "TriageTools : tools for partitioning and prioritizing analysis of high-throughput sequencing data," pp. 1–8, 2013.

[19] N. Maillet *et al.*, "Compareads: comparing huge metagenomic experiments." *BMC bioinformatics*, vol. 13 Suppl 1, no. Suppl 19, p. S10, Jan. 2012.

[20] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," vol. 13, pp. 422–426, 1970.

[21] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/9230594

[22] T. O. Delmont *et al.*, "Accessing the soil metagenome for studies of microbial diversity," *Applied and Environmental Microbiology*, vol. 77, no. 4, pp. 1315–1324, Feb 2011.

[23] F. Meyer *et al.*, "The metagenomics RAST server – a public resource for the automatic phylogenetic and functional analysis of metagenomes," *BMC bioinformatics*, vol. 9, no. 1, p. 386, Jan 2008.