Reproducibility of the paper "Whole-Genome Shotgun Metagenomic Sequencing Reveals Distinct Gut Microbiome Signatures of Obese Cats"

Jules Kreuer[1,✉], **Dario Eltzner**[1,✉], **and Philip Höllein**[1,✉]

[1]Bioinformatics Seminar BIOINF4240, Algorithms in Bioinformatics, Eberhard Karls Universität Tübingen

Obesity is an issue that deserves investigation even in domestic cats. Here we try to reproduce the whole-genome shotgun metagenomic study performed by Ma et al. [17]. Using a subset of the available data, we reconstruct a reference metagenome from both obese and lean cats and use this for whole-genome-based taxonomic community profiling. In this, we are able to reproduce the main features of the discovered community profile.

Furthermore, we assess the degree to which the authors have made a faithful reproduction of their results possible and identify minor – but often overlooked – aspects of providing open data and methodological details that are missing from the paper as published.

The complete pipeline and data from intermediate steps can be found on GitHub

Reproducibility Study | Bioinformatics Pipeline | Metagenome | Obesity | Cats

Correspondence: ***jules.kreuer@uni-tuebingen.de***

Introduction

In the publication 'Whole-Genome Shotgun Metagenomic Sequencing Reveals Distinct Gut Microbiome Signatures of Obese Cats' by Ma et al. a pipeline of openly available bioinformatics tools is put together by the authors. It is used to compute the relative abundance of bacteria in the microbiome of obese and lean cats. While the biological findings and the analysis based on them are remarkable, we are more interested in the reproducibility of the computational analysis.

A. Motivation. As overweight and obesity in domestic cats is a growing problem affecting around 45% of cats [17], and the health of our companions are essential to us, the authors decided to study the feline gut microbiome and create the first high-quality data-set for further analysis. Especially the lack of licensed drugs and the strong effect of the gut microbiome on many aspects of disease physiology in humans and other mammals indicated by other studies [17] drove the author's interest.

B. Goal. Although other studies have looked at the feline microbiome, none have reached the gold standard methodologically at the time of publication.

In particular, the authors highlight that preexisting studies all used 16S amplicon sequencing instead of whole genome shotgun sequencing. Whole-genome sequencing however is able to identify significantly more bacterial species with a higher specificity [20].

Furthermore, among previous studies some used samples from client-owned cats in household environments or took faecal samples from litter boxes. The latter may be contaminated, and the microbiota composition may shift after the faecal sample has left the gut.

This has spurred the authors Ma et al. to pay particular attention to the collection of samples, laying the foundation for a high-quality dataset and thus for high-quality analysis. This analysis includes community profiling of bacterial organisms and functional composition and pathway analysis using the KEGG pathway database.

C. Our Motivation. Reproducibility is arguably the most important hallmark of any mature scientific endeavour. With a lingering reproducibility crisis in life science research, this issue has become an object of sharp attention for the scientific community [22]. The unparalleled increase in life science research and the volumes of data produced, which ensued the ushering in of the age of genomics, have not yet been met by the capacity to reproduce the made discoveries.
For one, this is an issue of human resources and the unavoidable shifts of scientific attention that characterise a rapidly moving scientific field. However, improvements in methodological standards and research procedures also have been argued to be necessary and many strides have been made in improving the situation.
This includes the provisioning of open data and adherence to standards related to it, diligence in detailing used protocols, and availability of sufficient information about used proprietary technology, which is necessary for a meaningful analysis.

Bioinformatics as a discipline arguably has, for a long time already, maintained a high standard in these regards, with the majority of popular tools being fully open source and thereby open to scrutiny, and widespread availability of data sets in non-proprietary databases. However, there are still barriers to the widespread adoption of a culture of reproduction,

as well as minor issues in protocol reporting that warrant attention.
While addressing the issue of scientific reproduction generally having a lower publication value than new discovery, hence attracting fewer resources, it is far from having an obvious solution, and remaining technical barriers seem to be far easier to resolve.
Therefore, our main focus here is not on the statistical analysis performed by the authors, but rather on the technicalities of their pipeline. In particular, the choice of parameters in bioinformatics tools usually has an immense effect on the results and draws just as heavily from heuristic reasoning and experience, as from the understanding of the algorithmic strategy the tools implement.

D. Results. We used the pipeline reported by the authors as faithfully as possible. In particular, this pertains to the exact versions of used tools and their parameters.
In doing so, we found that the central findings of the community profiling of the study can in fact be reproduced rather precisely without excessive guesswork, even by novice bioinformaticians like ourselves. To illustrate we include a brief comparison of our results with those of the paper.

However, we also took notice of the fact that the authors by and large did not report the parameters they used. While the goal of the paper – providing a robust data set for subsequent usage – has undoubtedly been reached, not all of their choices and reasoning are open for scrutiny.

We finally propose an obvious and also quite well-adopted solution to this issue – publishing the exact pipeline as scripts – and discuss some of its shortcomings, as well as discuss some issues around the availability of tooling.

Methods and Comparison

This section will provide insights into the author's methods, the reproducibility of certain steps, and the choices we made, if a deviation from the original pipeline was necessary. We append the pipeline scripts we wrote in our efforts to reproduce the paper, which also contain our parameters, and published everything including data from intermediate steps on GitHub: `https://github.com/not-a-feature/reproducibility-cat-metagenome-analysis/`

A simplified overview of the pipeline used by the authors can be found in the supplementary notes 6.

E. Sample Collection. To counteract the previously mentioned problems with sample collection, the samples were taken directly from the colon using a plastic loop. Furthermore, the cats were laboratory animals, fed exclusively with lab-grade food, and didn't receive any anti or probiotics prior to the sample extraction. Using a standard protocol $1.5-2\mu$g of DNA was extracted and fragmented using ultrasound for a target insert size of 500bp. These fragments were prepared with the NEBNext Ultra II DNA Library Prep Kit, a metagenomic WGS library, and sequenced using an Illumina NovaSeq6000.

F. Data Availability. For archival and reproducibility purposes these paired-end reads of length 150 were uploaded to the NCBI Sequence Read Archive (SRA) under the accession number PRJNA758898. The metagenome assembly has been made available at NCBI Genome DataHub under the accession GCA_022675345.1

G. Reproduction Dataset. To keep the computation and download time low, we reduced the number of samples for the upcoming analysis to a total of four:
Group Lean Cat:

- SRR16235397/9Z113
- SRR16235398/9Z110

Group Obese Cat:

- SRR16235395/D001
- SRR16235406/H001

Information about the other samples and summary statistics computed by the study authors can be seen in Table 2.

Using the `SRA-Tools` v2.11.0 [21] the reads were downloaded and extracted into two files per sampleA. The raw reads presented themselves in excellent quality with mean quality scores mostly above 35 1:

Fig. 1. Per base sequence quality of the 2×4 raw reads. Analysed with FastQC [1], summarised with MultiQC [7].

H. Input Data and Trimming / Paired-end Merging. The authors used `PEAR` [25] version 0.9.11 to merge paired-end reads and `Trimmomatic` [2] [3] version 0.36 to clean up adapter sequences and low-quality base-calls B. The exact adapter sequences used for trimming were not reported. Therefore, after a manual inspection of the reads, we used the TruSeq3-PE adapters.

I. Cat / Viral Filtering. To remove the host and viral contamination of the sample the Burrows-Wheeler Aligner (`BWA`) version 0.7.17-r1188 and `SAMtools` version 1.6 was used to align reads against the feline reference genome GCF_000181335.3 and against the concatenated viral genome database from NCBI [24]. Since the study parameters are not known, we chose what we considered default parameters. Sequences that aligned successfully were removed with the help of a self-written Bash / Python script and the use of `SAMtools` B A.

J. Metagenomic Reference. `Megahit` [16] [15] version 1.1.2 was used to assemble the remaining reads. This is the only tool for which the authors reported parameters, as "with default parameters". The authors mention they filtered short contigs of below 400bp, which we implemented with a Python script B. Contigs with more than 95% global identity were removed using `cd-hit-est` version 4.7 [8]. An overview of the resulting contigs-size-distribution can be found in figure 2.

K. Taxonomic Annotation. The authors used `Kaiju` version 1.7.3 for taxonomic assignment with the NCBI taxonomy [18].
Due to time constraints and repeated crashes of the indexing step of `Kaiju`, we opted for the faster tools `Diamond` version 2.0.15 [5] [4] and `Megan` version 6.21.7 [11] [10], using the `Annotree Database` [9] for the taxonomy. This database allows a 2-fold speedup over the classic `NCBI-nr` protein database while having slightly increased assignment rates in metagenomic applications D.

L. Relative Frequency Assessment. The pre-processed (i.e. read-merged, filtered and trimmed) reads were then aligned against the assembled metagenomic reference. The authors do not explicitly mention which tool they used for alignment, so we assumed they used `BWA` like in the previous step E. Using these alignments, relative frequencies of taxonomic assignments per read set were computed. For this, we implemented a Bash / Python script C.

Results

We chose four of the 16 original samples, two of each obese and normal cat. We thereby reduced the original 1.8 billion 150bp reads down to 560 million reads. After trimming and paired-end read merging, we ended up with 524 million reads with varying lengths, 306 million of which remained unmerged. We then aligned all reads against the reference cat genome and the viral genomes and removed all aligned reads from the samples. Table 1 shows the comparison between the number of aligned reads for the study and our data. In total, 22.1% of our reads aligned against the cat genome, and 0.08% against the viral genome, leaving a total of 408 million reads across the four samples. We report a much higher alignment rate against both genomes, presumably due to different parameters of the `BWA`.

Table 1. Percentage contamination measured as the fraction of total reads aligned to cat and viral genomes for the study and our data

Percentage contamination	Study	Our data
Reads aligned to cat genome	6.31%	22.1%
Reads aligned to viral genome	0.05%	0.08%

After removing these non-microbial reads, we performed *de novo* metagenomic assembly using the four samples pooled into one. This assembly contained 498.938 contigs with a total length of 679,5 million base pairs. The longest contig was 375.251 base pairs. Compared with the 961 million base pairs distributed on 355.573 contigs of the assembly by Ma et al., our shorter assembly does fit expectations insofar, as it is significantly shorter and more scattered than the reference due to the lower sample size. A higher sample size would likely not increase the assembly size linearly but still lead to a noticeable increase in size.
The contig size distribution before and after filtering and applying `cd-hit-est` is presented in figure 2. While having many outliers – the longest contig being 370 kbp – the median and first/third quartile are much shorter at 500-1500 bp. The decreased coverage due to our reduced sample size may be a major cause of this comparatively low-quality assembly.

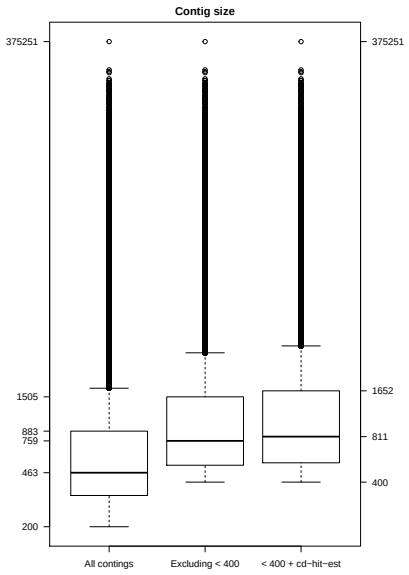

Fig. 2. Contig size after assembly to the metagenomic reference genome.

After performing taxonomic binning on the reference metagenome, we individually aligned our samples back against the reference genome and counted the read alignments against a classified contig. This way we were able to map 194.951.876 reads of lean cats and 100.014.404 reads of obese cats to phylum assignments. We then divided each aligned taxonomy by the total number of aligned reads per sample, giving us a measure of the relative abundance. Note that this is in line with how the authors report their abundance calculation. Figure 3 shows three of the top five most highly abundant taxa in all our samples, compared with the study data. We can see, that we arrive at similar relative abundances as the study. However, we did not perform the Mann-Whitney U-test, since comparing just 2 samples between both groups severely lacks statistical power.

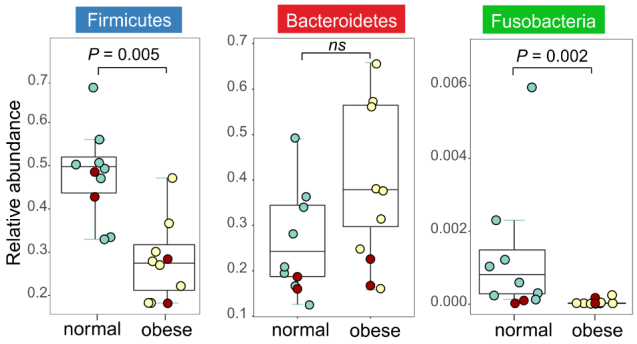

Fig. 3. Relative abundance boxplots of *firmicutes*, *bacteriodetes* and *fusobacteria* for normal and obese cats. Green and yellow dots mark the exact values of the study, red points mark our data. Mann-Whitney U tests were executed on the subgroups normal and obese for the study data points, showing significant group differences for *firmicutes* and *Fusobacteria*

We compare the average relative abundance of the top 5 phyla for our 2 samples per group with the relative abundances reported in the study. Figure 4 shows a stacked bar chart of the top 5 most abundant phyla. As seen in Figure 3, this confirms the higher relative abundance of *firmicutes* in normal cats, but also a higher *actinobacter* abundance in obese cats, and an equal *bacteriodetes* abundance.

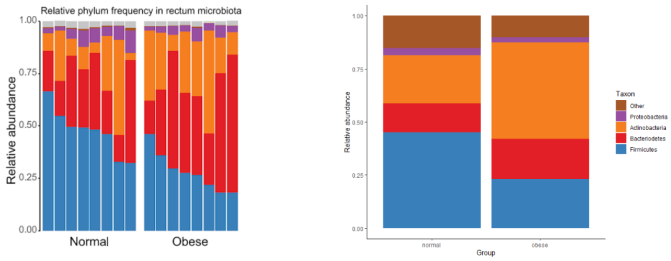

Fig. 4. Stacked bar charts of the relative abundance of the top 5 phyla found in normal and obese cats for study (left) and our data (right, averaged)

This allows a closer look at the *firmicutes* to *bacteriodetes* ratio across groups. This ratio has often been associated with obesity (with some papers observing the same tendency [23], and some the reverse [12]), where *bacteriodetes* become the most dominant taxon among obese individuals while remaining lower than *firmicutes* in the normal population. Figure 5 shows the comparison of this ratio for the study and our data. While in the study the ratio drops slightly below 1 for obese cats, our data shows a ratio of slightly above 1. This is well within the measure of uncertainty of the original data, but more data points would increase interpretability.

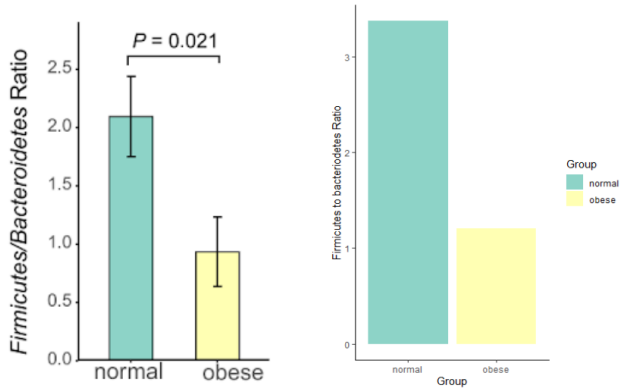

Fig. 5. Average ratio of *firmicutes* to *bacteriodetes* in normal and obese cats for study (left) and our data (right)

We did not carry out any further (statistical) analyses as the data was insufficient due to the reduced number of samples. However, since we achieved similar relative abundances for our samples compared with the study, all further analyses relying solely on statistical methods and higher sample size would have also been possible.

Reproducibility. The authors reported parameters only for the assembler they used – however, there is still some ambiguity there ("with default parameters"). Information on what pre-processing (demultiplexing, trimming etc.) was already performed on the reads that are now available via SRA was not provided. Finally, the exact adapter sequences used for trimming were not reported.

Due to time constraints, were not able to complete the full assembly of the metagenome reference. Therefore, we have no way of telling how the full result would have compared to the results of the study. However, given our discrepancy in the number of filtered reads alone, it is unlikely that the result is as close to the authors' results, as it could have been, had they published their parameters.

Many authors have started to publish pipeline scripts as open-source code (e.g. [19]). While this is undoubtedly an improvement, there are still some problems with it. Pipelines like these are usually created through an iterative process of implementation, debugging and quality control. There is some chance that a script will deviate in some way from what the authors actually did with their data. Scripts also need to be published in a reliable way. Without comprehensive documentation of the tool versions, operating system and even hardware, the raw code of a script is not sufficient. In addition, the specific dependencies of tools (at least through the usual distribution channels) often require some level of environment management that should be included in the source code.

While most of the tools used by the authors are available through established package managers - especially the `bioconda` channel of the `conda` package manager - some tools and versions were not immediately available. For exam-

ple, version 4.7 of `cd-hit-est` is not available via `conda` and is skipped as a release on GitHub. However, we were able to either acquire these as binaries from university websites or build them from source.

Although this seems to be a negligible problem, we mention it because – in our personal experience – it is not always guaranteed that the code will be available or compilable in the future. This is a notorious problem with Python build systems, where people tend to specify open version ranges for dependencies that then turn out to be incompatible with newer versions that are released.

Standard software industry practice for use cases comparable to scientific software production (i.e. open source development) is to deploy code either via package managers or as containers, or to publish specifications for build environments in a fixed version, either via build containers or environment specification exports (i.e. `pip list` or `conda env export`. It seems desirable that not only the data manipulation performed by a study is fully documented, but also that the code executed is available and ready for use.

M. Conclusion. Problems in the reproducibility of biomedical studies are not uncommon. Although the authors Ma et al. of the paper 'Whole-Genome Shotgun Metagenomic Sequencing Reveals Distinct Gut Microbiome Signatures of Obese Cats' put a lot of emphasis on the sample collection methodology, there are difficulties in reproducing the results due to a lack of information about the computational pipeline. However, there was still enough information available to infer the missing parts and compute reasonably similar results.

We have tried to highlight the missing parts, their impact on the results and described some best practices for reproducibility, such as publishing all parameters and package versions. The recreated pipeline was put together using several bash and python scripts which can be found including some intermediate results in the appendix and on GitHub.

N. Author contribution. All authors contributed equally.

Bibliography

References

[1] Simon Andrews. *FastQC*. Comp. software. Version 0.11.9. Jan. 8, 2020. URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/.

[2] Anthony M. Bolger, Marc Lohse, and Bjoern Usadel. "Trimmomatic: a flexible trimmer for Illumina sequence data". In: *Bioinformatics* 30.15 (Apr. 2014), pp. 2114–2120. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btu170. eprint: https://academic.oup.com/bioinformatics/article-pdf/30/15/2114/17143152/btu170.pdf. URL: https://doi.org/10.1093/bioinformatics/btu170.

[3] Tony Bolger and Björn Usadel. *Trimmomatic*. Comp. software. Version 0.39. Jan. 22, 2021. URL: https://github.com/usadellab/Trimmomatic.

[4] Benjamin Buchfink. *Diamond*. Comp. software. Version 2.0.15. Apr. 21, 2022. URL: https://github.com/bbuchfink/diamond.

[5] Benjamin Buchfink, Klaus Reuter, and Hajk-Georg Drost. "Sensitive protein alignments at tree-of-life scale using DIAMOND". In: *Nature Methods* 18.4 (Apr. 2021), pp. 366–368. DOI: 10.1038/s41592-021-01101-x. URL: http://dx.doi.org/10.1038/s41592-021-01101-x.

[6] Simon Duerr. *Bioicons - high quality science illustrations*. URL: https://bioicons.com/.

[7] Philip Ewels et al. "MultiQC: summarize analysis results for multiple tools and samples in a single report". In: *Bioinformatics* 32.19 (June 16, 2016), pp. 3047–3048. DOI: 10.1093/bioinformatics/btw354. URL: http://dx.doi.org/10.1093/bioinformatics/btw354.

[8] Limin Fu et al. "CD-HIT: accelerated for clustering the next-generation sequencing data". In: *Bioinformatics* 28.23 (Oct. 2012), pp. 3150–3152. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts565. eprint: https://academic.oup.com/bioinformatics/article-pdf/28/23/3150/18529929/bts565.pdf. URL: https://doi.org/10.1093/bioinformatics/bts565.

[9] Anupam Gautam et al. "Using AnnoTree to Get More Assignments, Faster, in DIAMOND+MEGAN Microbiome Analysis". In: *mSystems* 7.1 (2022), e01408–21. DOI: 10.1128/msystems.01408-21. eprint: https://journals.asm.org/doi/pdf/10.1128/msystems.01408-21. URL: https://journals.asm.org/doi/abs/10.1128/msystems.01408-21.

[10] Daniel Huson. *Megan*. Comp. software. Version 6.21.7. 2021. URL: https://www.wsi.uni-tuebingen.de/lehrstuehle/algorithms-in-bioinformatics/software/megan6/.

[11] Daniel H. Huson et al. "MEGAN Community Edition - Interactive Exploration and Analysis of Large-Scale Microbiome Sequencing Data". In: *PLOS Computational Biology* 12.6 (June 21, 2016). Ed. by Timothée Poisot, e1004957. DOI: 10.1371/journal.pcbi.1004957. URL: http://dx.doi.org/10.1371/journal.pcbi.1004957.

[12] Reiner Jumpertz et al. "Energy-Balance Studies Reveal Associations between Gut Microbes, Caloric Load, and Nutrient Absorption in Humans". In: *The American Journal of Clinical Nutrition* 94.1 (July 1, 2011), pp. 58–65. ISSN: 0002-9165. DOI: 10.3945/ajcn.110.010132. URL: https://doi.org/10.3945/ajcn.110.010132 (visited on 09/29/2022).

[13] Jules Kreuer. *not-a-feature/fastq: v1.2.2 hash and py3.7 support*. Version v1.2.2. Sept. 2022. DOI: 10.5281/zenodo.7110280. URL: https://doi.org/10.5281/zenodo.7110280.

[14] Jules Kreuer. *not-a-feature/miniFASTA: v2.4.1 add py3.7 support and doi*. Version v2.4.1. Sept. 2022. DOI: 10.5281/zenodo.7110267. URL: https://doi.org/10.5281/zenodo.7110267.

[15] Dinghua Li. *MEGAHIT*. Comp. software. Version 1.1.2. Aug. 1, 2017. URL: https://github.com/voutcn/megahit.

[16] Dinghua Li et al. "MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph". In: *Bioinformatics* 31.10 (Jan. 2015), pp. 1674–1676. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btv033. eprint: https://academic.oup.com/bioinformatics/article-pdf/31/10/1674/17085710/btv033.pdf. URL: https://doi.org/10.1093/bioinformatics/btv033.

[17] Xiaolei Ma et al. "Whole-Genome Shotgun Metagenomic Sequencing Reveals Distinct Gut Microbiome Signatures of Obese Cats". In: *Microbiology Spectrum* 10.3 (2022), e00837–22. DOI: 10.1128/spectrum.00837-22. eprint: https://journals.asm.org/doi/pdf/10.1128/spectrum.00837-22. URL: https://journals.asm.org/doi/abs/10.1128/spectrum.00837-22.

[18] Peter Menzel. *Kaiju*. Comp. software. Version 1.7.3. Jan. 16, 2020. URL: https://github.com/bioinformatics-centre/kaiju.

[19] Daniel M. Portik, C. Titus Brown, and N. Tessa Pierce-Ward. "Evaluation of Taxonomic Profiling Methods for Long-Read Shotgun Metagenomic Sequencing Datasets". In: (Feb. 2, 2022), p. 2022.01.31.478527. DOI: 10.1101/2022.01.31.478527. URL: https://www.biorxiv.org/content/10.1101/2022.01.31.478527v1 (visited on 09/19/2022).

[20] Ravi Ranjan et al. "Analysis of the microbiome: Advantages of whole genome shotgun versus 16S amplicon sequencing". In: *Biochemical and Biophysical Research Communications* 469.4 (Jan. 2016), pp. 967–977. DOI: 10.1016/j.bbrc.2015.12.083. URL: http://dx.doi.org/10.1016/j.bbrc.2015.12.083.

[21] *SRA tools*. Comp. software. Version 2.11. Mar. 15, 2021. URL: https://github.com/ncbi/sra-tools.

[22] Charlotte Stoddart. "Is there a reproducibility crisis in science?" In: *Nature* (May 25, 2016). DOI: 10.1038/d41586-019-00067-3. URL: http://dx.doi.org/10.1038/d41586-019-00067-3.

[23] Peter J. Turnbaugh et al. "A Core Gut Microbiome in Obese and Lean Twins". In: *Nature* 457.7228 (7228 Jan. 2009), pp. 480–484. ISSN: 1476-4687. DOI: 10.1038/nature07540. URL: https://www.nature.com/articles/nature07540 (visited on 09/29/2022).

[24] David L. Wheeler et al. "Database Resources of the National Center for Biotechnology Information". In: *Nucleic Acids Research* 28.1 (Jan. 1, 2000), pp. 10–14. ISSN: 0305-1048. DOI: 10.1093/nar/28.1.10. URL: https://doi.org/10.1093/nar/28.1.10 (visited on 09/28/2022).

[25] Jiajie Zhang et al. "PEAR: A Fast and Accurate Illumina Paired-End reAd mergeR". In: *Bioinformatics* 30.5 (Mar. 1, 2014), pp. 614–620. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btt593. pmid: 24142950. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3933873/ (visited on 09/19/2022).

O. Study database and chosen samples. The study consisted of a total of 16 samples, 8 lean, and 8 obese cats. Table 2 shows summary statistics of those samples computed by the study authors, and the samples we chose are marked in yellow.

Table 2. Summary statistics of all 16 cat samples computed by the study authors

Animal ID	total number of reads	total yield (Gbp)	% adapters & low quality reads	% host sequences	% viral sequences	% read alignment
111468	224,309,608	33.65	1.26%	1.75%	0.06%	96.54%
111473	252,204,562	37.83	0.78%	0.85%	0.05%	97.76%
81392	68,992,950	10.35	1.65%	19.62%	0.04%	78.82%
111030	67,304,938	10.10	1.05%	3.26%	0.05%	95.31%
111041	62,165,458	9.32	1.49%	3.14%	0.05%	95.31%
9Z110	177,167,802	26.58	1.68%	5.54%	0.04%	92.28%
9Z113	187,101,498	28.07	1.23%	3.43%	0.06%	94.74%
9Z116	71,831,380	10.77	1.84%	9.91%	0.05%	88.36%
D001	114,565,502	17.18	2.74%	16.53%	0.03%	81.42%
F001	88,239,704	13.24	3.37%	69.19%	0.01%	27.67%
G001	96,951,866	14.54	3.81%	35.50%	0.02%	61.68%
H001	81,550,038	12.23	2.44%	0.21%	0.04%	98.16%
I001	88,388,090	13.26	2.67%	2.09%	0.03%	95.74%
J001	93,732,658	14.06	3.11%	21.16%	0.02%	76.28%
K001	54,533,852	8.18	2.30%	2.66%	0.03%	95.46%
L001	77,921,032	11.69	3.89%	48.57%	0.02%	48.22%

P. Pipeline overview. Figure 6 shows a simplified overview of the pipeline used by the authors Ma et al. The analysis based on relative frequency was excluded.

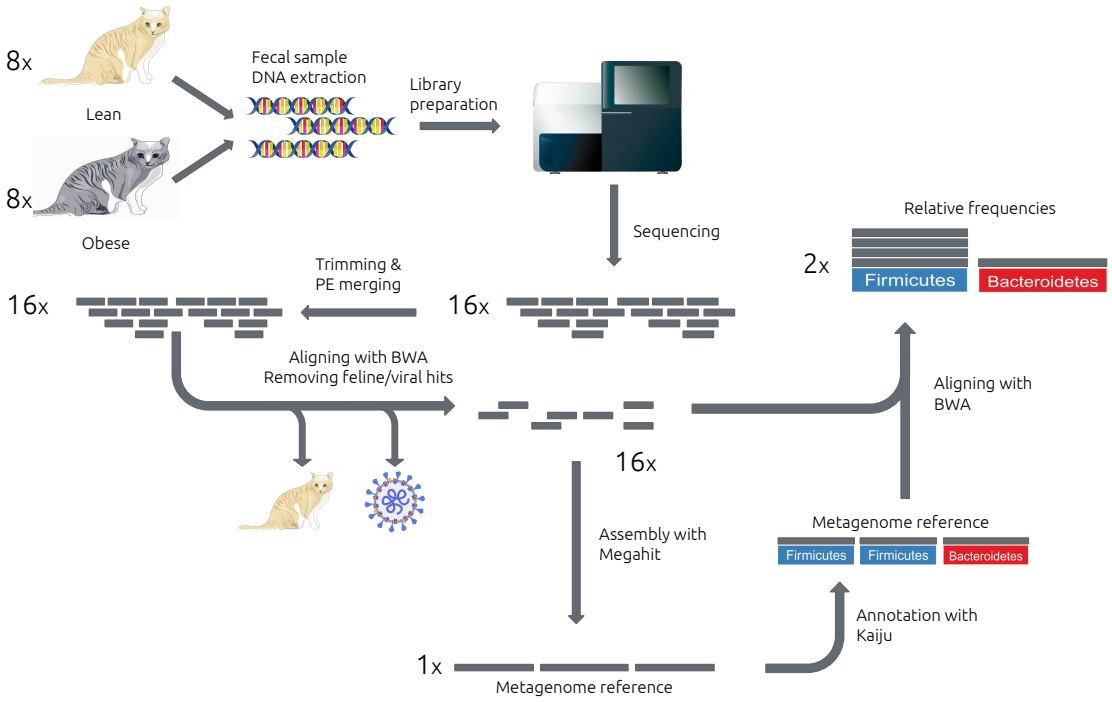

Fig. 6. Simplified pipeline overview. Icons from BioIcons.com [6]
Cat by Servier (CC-BY 3.0), Virus by Hanna Vega, Sequencer by DBCL (CC-BY 4.0)

Supplementary Note 1: Bash Pipelines

A. download_data.sh. This script downloads and prepare all necessary files from the NCBI archives. View on GitHub.

```
#!/bin/bash
set -e
# Takes 4 sample ids as input
INPUT_1=$1
ID_1=$(basename $INPUT_1)
INPUT_2=$2
ID_2=$(basename $INPUT_2)
INPUT_3=$3
ID_3=$(basename $INPUT_3)
INPUT_4=$4
ID_4=$(basename $INPUT_4)

WS=$5
VIRAL_REFERENCE_GENOME="$WS/data/bwa/viral_merged.faa"

# Raw Reads
# Download 4 of the 16 raw reads from the SRA.
fastq-dump $ID_1 --split-3 --outdir $WS/data/raw_reads/
echo "Raw read download: 1/4 complete"
fastq-dump $ID_2 --split-3 --outdir $WS/data/raw_reads/
echo "Raw read download: 2/4 complete"
fastq-dump $ID_3 --split-3 --outdir $WS/data/raw_reads/
echo "Raw read download: 3/4 complete"
fastq-dump $ID_4 --split-3 --outdir $WS/data/raw_reads/
echo "Raw read download: 4/4 complete"

# Feline reference genome
wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/181/335/
GCF_000181335.3_Felis_catus_9.0/GCF_000181335.3_Felis_catus_9.0_genomic.fna.gz \
  -O $WS/data/bwa/GCF_000181335.3_Felis_catus_9.0_genomic.fna.gz
echo "Feline reference download complete"

# Viral reference genome
wget ftp://ftp.ncbi.nlm.nih.gov/refseq/release/viral/viral.1.genomic.gbff.gz \
  -O $WS/data/bwa/viral.1.genomic.faa.gz
wget ftp://ftp.ncbi.nlm.nih.gov/refseq/release/viral/viral.2.genomic.gbff.gz \
  -O $WS/data/bwa/viral.2.genomic.faa.gz
wget ftp://ftp.ncbi.nlm.nih.gov/refseq/release/viral/viral.3.genomic.gbff.gz \
  -O $WS/data/bwa/viral.3.genomic.faa.gz
wget ftp://ftp.ncbi.nlm.nih.gov/refseq/release/viral/viral.4.genomic.gbff.gz \
  -O $WS/data/bwa/viral.4.genomic.faa.gz
echo "Viral reference download complete"

# Unpack and merge
zcat $WS/data/bwa/viral.*.faa.gz > $VIRAL_REFERENCE_GENOME
# Index for further use
bwa index $VIRAL_REFERENCE_GENOME

# Adapters for trimmomatic
echo ">PrefixPE/1" >> $WS/TruSeq3-PE.fa
echo "TACACTCTTTCCCTACACGACGCTCTTCCGATC" >> $WS/TruSeq3-PE.fa
echo ">PrefixPE/2" >> $WS/TruSeq3-PE.fa
echo "GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT" >> $WS/TruSeq3-PE.fa
```

B. filer_raw_reads.sh. This script takes sample id (eg.: SRR16235395) as input, and aligns the reads against the feline and viral reference genome. Matches against those references are filtered out.
View on GitHub.

```
#!/bin/bash
set -e

# Read name
INPUT=$1
BASENAME=$(basename $INPUT)
WS=$2

# Path to PEAR binary
PEAR_BINARY=$WS/pipeline/pear-0.9.11-linux-x86_64/bin/pear

INPUT_1="$WS/data/raw_reads/${BASENAME}_1.fastq"
INPUT_2="$WS/data/raw_reads/${BASENAME}_2.fastq"

BWA_OUTPUT_DIR="$WS/data/bwa/aligned"
# Path to python filter script
FILTER_SCRIPT="$WS/pipeline/exclude_from_fq.py"

PEAR_FORWARD="$WS/data/pear_reads/$BASENAME.unassembled.forward.fastq"
PEAR_REVERSE="$WS/data/pear_reads/$BASENAME.unassembled.reverse.fastq"
PEAR_MERGED="$WS/data/pear_reads/$BASENAME.assembled.fastq"

# Path to Reference genome
CAT_REFERENCE_GENOME="$WS/data/bwa/GCF_000181335.3_Felis_catus_9.0_genomic.fna.gz"
VIRAL_REFERENCE_GENOME="$WS/data/bwa/viral_merged.faa"

SAM_UNMERGED="$BWA_OUTPUT_DIR/$BASENAME.pe.sam"
SAM_MERGED="$BWA_OUTPUT_DIR/$BASENAME.merged.sam"

INDEX_FILE_UNMERGED="$WS/data/extracted/$BASENAME.pe.index"
INDEX_FILE_MERGED="$WS/data/extracted/$BASENAME.merged.index"

NOCAT_FORWARD="$WS/data/extracted/$BASENAME.forward.nocat.fq"
NOCAT_REVERSE="$WS/data/extracted/$BASENAME.reverse.nocat.fq"
NOCAT_MERGED="$WS/data/extracted/$BASENAME.merged.nocat.fq"

NOCAT_SAM_UNMERGED="$BWA_OUTPUT_DIR/$BASENAME.nocat.pe.sam"
NOCAT_SAM_MERGED="$BWA_OUTPUT_DIR/$BASENAME.nocat.merged.sam"

TRIMMED_PATH="$WS/data/trimmed"

NOCAT_INDEX_FILE_UNMERGED="$WS/data/extracted/$BASENAME.nocat.pe.index"
NOCAT_INDEX_FILE_MERGED="$WS/data/extracted/$BASENAME.nocat.merged.index"

# trimming reads
PAIRED_TRIMMED_FORWARD=$TRIMMED_PATH/"${BASENAME}.ptf.fastq"
UNPAIRED_TRIMMED_FORWARD=$TRIMMED_PATH/"${BASENAME}.utf.fastq"
PAIRED_TRIMMED_REVERSE=$TRIMMED_PATH/"${BASENAME}.ptr.fastq"
UNPAIRED_TRIMMED_REVERSE=$TRIMMED_PATH/"${BASENAME}.utr.fastq"

UNPAIRED_MERGED=$TRIMMED_PATH/"${BASENAME}.um.fastq"

echo "Trimming reads."
trimmomatic PE -threads 60 \
            -trimlog $WS/"${BASENAME}.trim.log" \
            $INPUT_1 $INPUT_2 \
            $PAIRED_TRIMMED_FORWARD $UNPAIRED_TRIMMED_FORWARD \
            $PAIRED_TRIMMED_REVERSE $UNPAIRED_TRIMMED_REVERSE \
            ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 MINLEN:36

# pear reads

echo "PEAR reads"
$PEAR_BINARY -f $PAIRED_TRIMMED_FORWARD \
             -r $PAIRED_TRIMMED_REVERSE \
             -o $WS/pear_reads/$BASENAME \
             -j 60 # threads

# merging PEAR and trimmomatic stuff

cat $PEAR_MERGED $UNPAIRED_TRIMMED_FORWARD $UNPAIRED_TRIMMED_REVERSE > $UNPAIRED_MERGED

# filter cat


echo "Map vs. cat reference."
echo "Filtering cats. 1/2"
bwa mem -t 60 \
   $CAT_REFERENCE_GENOME \
   $PEAR_FORWARD $PEAR_REVERSE \
   -o "$BWA_OUTPUT_DIR/$BASENAME.cataligned.pe.bam"
```

```
echo "Filtering cats. 2/2"
bwa mem -t 60 \
   $CAT_REFERENCE_GENOME \
   $UNPAIRED_MERGED \
   -o "$BWA_OUTPUT_DIR/$BASENAME.cataligned.merged.bam"

echo "Extracting headers."
samtools view -Shf 4 -h $BWA_OUTPUT_DIR/"$BASENAME.cataligned.pe.bam" > $SAM_UNMERGED
samtools view -Shf 4 -h $BWA_OUTPUT_DIR/"$BASENAME.cataligned.merged.bam" > $SAM_MERGED

echo "Creating extracted ID list."
grep -v "@" $SAM_UNMERGED  | awk NF=1 > $INDEX_FILE_UNMERGED
grep -v "@" $SAM_MERGED  | awk NF=1 > $INDEX_FILE_MERGED

echo "Filtering reads."
python3 $FILTER_SCRIPT -l $INDEX_FILE_UNMERGED -fq $PEAR_FORWARD -o $NOCAT_FORWARD
python3 $FILTER_SCRIPT -l $INDEX_FILE_UNMERGED -fq $PEAR_REVERSE -o $NOCAT_REVERSE
python3 $FILTER_SCRIPT -l $INDEX_FILE_MERGED -fq $UNPAIRED_MERGED -o $NOCAT_MERGED

# filter virus

echo "Map vs. virus reference."
echo "Filtering viruses. 1/2"
bwa mem -t 60 $VIRAL_REFERENCE_GENOME $NOCAT_FORWARD $NOCAT_REVERSE \
    -o $BWA_OUTPUT_DIR/"$BASENAME.virusaligned.nocat.pe.bam"
echo "Filtering viruses. 2/2"
bwa mem -t 60 $VIRAL_REFERENCE_GENOME $NOCAT_MERGED \
    -o $BWA_OUTPUT_DIR/"$BASENAME.virusaligned.nocat.merged.bam"

echo "Extracting headers."
samtools view -Shf 4 -h $BWA_OUTPUT_DIR/"$BASENAME.virusaligned.nocat.pe.bam" \
    > $NOCAT_SAM_UNMERGED
samtools view -Shf 4 -h $BWA_OUTPUT_DIR/"$BASENAME.virusaligned.nocat.merged.bam" \
    > $NOCAT_SAM_MERGED

echo "Creating extracted ID list."
grep -v "@" $NOCAT_SAM_UNMERGED  | awk NF=1 > $NOCAT_INDEX_FILE_UNMERGED
grep -v "@" $NOCAT_SAM_MERGED  | awk NF=1 > $NOCAT_INDEX_FILE_MERGED

echo "Filtering reads."
python3 $FILTER_SCRIPT -l $NOCAT_INDEX_FILE_UNMERGED -fq $NOCAT_FORWARD \
    -o "$WS/data/extracted/$BASENAME.forward.filtered.fq"
python3 $FILTER_SCRIPT -l $NOCAT_INDEX_FILE_UNMERGED -fq $NOCAT_REVERSE \
    -o "$WS/data/extracted/$BASENAME.reverse.filtered.fq"
python3 $FILTER_SCRIPT -l $NOCAT_INDEX_FILE_MERGED -fq $NOCAT_MERGED \
    -o "$WS/data/extracted/$BASENAME.merged.filtered.fq"

echo "Create fastqc reports"
fastqc -t 20 $TRIMMED_PATH/*.fastq \
    || fastqc -t 20 $WS/data/pear_reads/*.fastq \
    || fastqc -t 20 $WS/data/extracted/*.fq \
    || true

chmod -R 771 $TRIMMED_PATH \
    || chmod -R 771 $BWA_OUTPUT_DIR \
    || chmod -R 771 $WS/data/extracted \
    || chmod -R 771 $WS/data/pear_reads \
    || true
```

C. megahit.sh. This will create a metagenomic reference using the preprocessed reads. View on GitHub.

```
#!/bin/bash
set -e

# Creates a metagenomic reference using megahit
# Takes 4 sample ids as input
WS=$5

INPUT_1=$1
ID_1=$(basename $INPUT_1)
INPUT_2=$2
ID_2=$(basename $INPUT_2)
INPUT_3=$3
ID_3=$(basename $INPUT_3)
INPUT_4=$4
ID_4=$(basename $INPUT_4)

megahit -1 $ID_1.forward.filtered.fq,$ID_2.forward.filtered.fq,$ID_3.forward.filtered.fq,$ID_4.forward.filtered.fq \
    -2 $ID_1.reverse.filtered.fq,$ID_2.reverse.filtered.fq,$ID_3.reverse.filtered.fq,$ID_4.reverse.filtered.fq \
    -r $ID_1.merged.filtered.fq,$ID_2.merged.filtered.fq,$ID_3.merged.filtered.fq,$ID_4.merged.filtered.fq \
    -t 48 \
    -o $WS/data/assembly
```

D. filter_cdhit_diamond_megan.sh. This script takes the megahit reference, filters it, and assigns a taxonomy to it using diamond and megan.
View on GitHub.

```
#!/bin/bash
set -e
# INPUT = reference fasta
INPUT=$1
WS=$2

PYTHON_SCRIPT="$WS/pipeline/filter_short_contigs.py"
PY_OUT_PATH="$WS/data/assembly/metagen_ref_filtered.fa"

# Path to cd-hit-est binary
# Or replace: CDHIT_BIN="cd-hit-est" when using the conda installation
CDHIT_BIN="$WS/pipeline/cdhit/cd-hit-est"

OUT_PATH="$WS/data/assembly/metagen_ref_filtered_cdhit.fa"
BWA_PATH="$WS/data/bwa_indiv/metagen_ref.fa"
CLASSIFICATION_PATH="$WS/data/classification"

# Filter short contigs
python3 $PYTHON_SCRIPT -fa $INPUT -l 400 -o $PY_OUT_PATH
# Filter redundant contigs
$CDHIT_BIN  -i $PY_OUT_PATH -o $OUT_PATH -c 0.95 -M 90000 -T 0

# Runs diamond and megan to assign taxonomy
DAA_OUT_PATH="$WS/data/diamond/metagen_ref_matches.daa"
DIAMONDDB="$WS/data/diamond/annotree.dmnd"
MEGANDB="$WS/pipeline/diamond/megan-mapping-annotree-June-2021.db"

diamond blastx -p 60 -d $DIAMONDDB -q $OUT_PATH -o $DAA_OUT_PATH --outfmt 100
daa-meganizer -i $DAA_OUT_PATH -mdb $MEGANDB
daa2info -i $DAA_OUT_PATH -r2c Taxonomy -n >> $CLASSIFICATION_PATH

# BWA index for individual alignment of samples against metagenomic reference
cp $OUT_PATH $BWA_PATH
```

E. bwa_metagenome.sh. Takes sample id as input, aligns the preprocessed reads against the metagenome reference.
View on GitHub.

```
#!/bin/bash
set -e

INPUT=$1
BASENAME=$(basename $INPUT)
WS=$2
BWA_INDEX=$3

METAGENOME="$WS/data/bwa_indiv/metagen_ref.fa"
RELFREQSCRIPT="$WS/pipeline/relative_frequencies.py"
TAXONOMY="$WS/data/classification"

BWA_OUTPUT_DIR="$WS/data/bwa_indiv"
CONTIGS_OUTPUT_DIR="$WS/data/contigs"
RELFREQ_OUTPUT_DIR="$WS/data/relfreqs"

PE_FORWARD="$WS/data/pear_reads/$BASENAME.unassembled.forward.fastq"
PE_REVERSE="$WS/data/pear_reads/$BASENAME.unassembled.reverse.fastq"

SAM_FILE="$CONTIGS_OUTPUT_DIR/$BASENAME.extracted.sam"
CONTIGS_FILE="$CONTIGS_OUTPUT_DIR/$BASENAME.contig.aligned"

# Bwa index only once
if [ "$BWA_INDEX" == "true" ]; then
    bwa index $METAGENOME
fi

#bwa mem
echo "Aligning against reference metagenom"
bwa mem -t 60 \
    $METAGENOME \
    $PE_FORWARD $PE_REVERSE \
    -o "$BWA_OUTPUT_DIR/$BASENAME.metaaligned.bam"

echo "Extracting aligned against reference contigs"
samtools view -h -o $SAM_FILE $BWA_OUTPUT_DIR/"$BASENAME.metaaligned.bam"

grep -v "@" $SAM_FILE | awk '{if ($3 != "*") print $3}' > $CONTIGS_FILE
echo "Calculating relative frequencies"
$RELFREQSCRIPT $TAXONOMY $CONTIGS_FILE > $RELFREQ_OUTPUT_DIR/"$BASENAME.relfreq"
```

Supplementary Note 2: Pyhton3 Scripts

A. exclude_from_fq.py. This script reads a fastq and a file containing all headers that should be kept. View on GitHub. This script uses the `fastq` package [13].

```
"""
@author: Jules Kreuer
@contact: jules.kreuer@uni-tuebingen.de
"""
import argparse
import fastq as fq

def main(LIST_PATH, FQ_PATH, OUT_PATH):
    print("Checking", FQ_PATH, "against", LIST_PATH)

    # Read header file to include
    headers = set()
    with open(LIST_PATH) as f:
        for l in f:
            headers.add(l.strip().lstrip("@"))

    if OUT_PATH is None:
        OUT_PATH = FQ_PATH.rsplit(".", 1)[0] + ".filtered.fq"
    print("Writing to", OUT_PATH)

    # Read fastq file
    fos = fq.read(FQ_PATH)

    # Filter reads
    fos = filter(lambda fo: fo.getHead().split()[0].lstrip("@") in headers, fos)
    fos = list(fos)

    fq.write(fos, OUT_PATH, "w")
    print("done")

if __name__ == "__main__":
    # Parse arguments
    parser = argparse.ArgumentParser(
        description='Remove reads that are not in header list')
    parser.add_argument('-l', "--list", action='store', dest='list',
        help='Specify path to list of headers to include', required=True)
    parser.add_argument('-fq', "--fastq", action='store', dest='fastq',
        help='Specify path to fastq file', required=True)
    parser.add_argument('-o', "--out", action='store', dest='out',
        help='Specify path to output file', required=False)

    args = parser.parse_args()
    LIST_PATH = args.list
    FQ_PATH = args.fastq
    OUT_PATH = args.out

    main(LIST_PATH, FQ_PATH, OUT_PATH)
```

B. filter_short_contigs.py. This script reads a fastq and a file containing all headers that should be kept. View on GitHub. This script uses the `miniFasta` package [14].

```
"""
@author: Jules Kreuer
@contact: contact@juleskreuer.eu
"""
import argparse
import miniFasta as mf

def main(MIN_LEN, FA_PATH, OUT_PATH):
    print("Checking", FA_PATH, ", min length", MIN_LEN)

    if OUT_PATH is None:
        OUT_PATH = FA_PATH.rsplit(".", 1)[0] + ".filtered.fq"
    print("Writing to", OUT_PATH)

    fas = mf.read(FA_PATH)

    # Filter reads
    fas = filter(lambda fa: MIN_LEN < len(fa), fas)
    fas = list(fas)

    mf.write(out, OUT_PATH, "a")
    print("done")

if __name__ == "__main__":
    # Parse arguments
    parser = argparse.ArgumentParser(
        description='Splits fastq files at pos 150')
    parser.add_argument('-fa', "--fasta", action='store', dest='fasta',
        help='Specify path to fasta file', required=True)
    parser.add_argument('-o', "--out", action='store', dest='out',
        help='Specify path to output file', required=False)
    parser.add_argument('-l', "--len", action='store', dest='len',
        help='Minimal length', required=True)

    args = parser.parse_args()
    MIN_LEN = int(args.len)
    FA_PATH = args.fasta
    OUT_PATH = args.out

    main(MIN_LEN, FA_PATH, OUT_PATH)
```

C. relative_frequencies.py. This script counts the metagenomic hits and computes the relative abundance. View on GitHub.

```
import sys
from collections import defaultdict
from group_assignment_data import *

phylum_counter = {}
tax_lookup = defaultdict(lambda: None)

with open(sys.argv[1]) as f:
    for line in f:
        dongle = line.split()
        tax_lookup[dongle[0].strip()] = dongle[1].strip()

for v in ["Actinobacteria", "Bacteriodetes", "Firmicutes", "Fusobacteria", "Proteobacteria", "Other"]:
    phylum_counter[v] = 0

total = 0
with open(sys.argv[2]) as f:
    for line in f:
        tax = tax_lookup[line.strip()]
        if tax:
            if tax in actinobacteria:
                phylum = "Actinobacteria"
            elif tax in bacteriodetes:
                phylum = "Bacteriodetes"
            elif tax in firmicutes:
                phylum = "Firmicutes"
            elif tax in fusobacteria:
                phylum = "Fusobacteria"
            elif tax in proteobacteria:
                phylum = "Proteobacteria"
            else:
                phylum = "Other"
            total = total + 1
            phylum_counter[phylum] = phylum_counter[phylum] + 1
for k, v in sorted(phylum_counter.items(), key=lambda x: x[1], reverse=True):
    print(f"{k} {v} {v / total}")
```