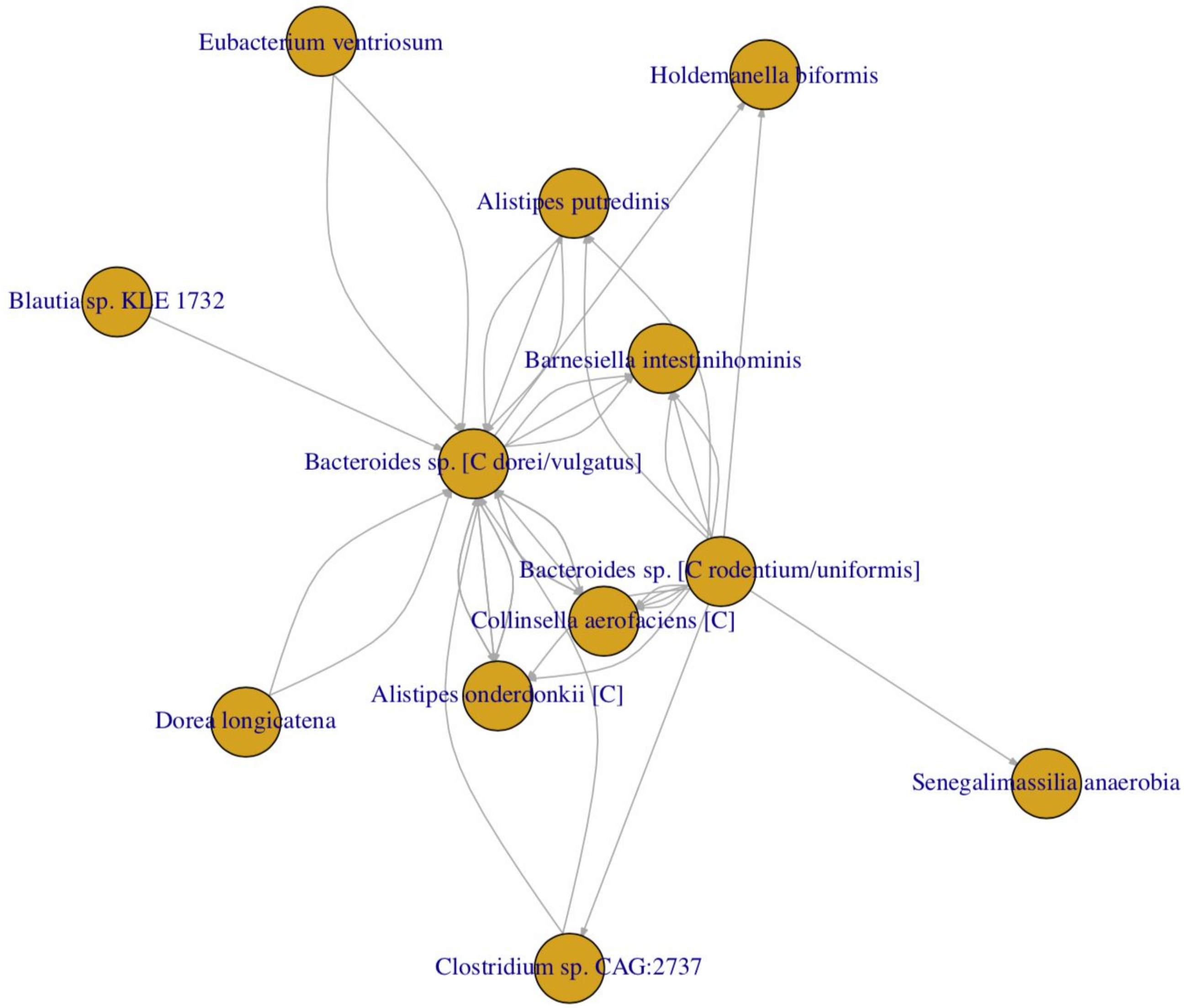
Exploring Social Interactions in Microbial Communities of the Human Gut


Eubacterium ventriosum
Holdemanella biformis
Alistipes putredinis
Blautia sp. KLE 1732
Barnesiella intestinihominis
Bacteroides sp. [C dorei/vulgatus]
Bacteroides sp. [C rodentium/uniformis]
Collinsella aerofaciens [C]
Alistipes onderdonkii [C]
Dorea longicatena
Senegalimassilia anaerobia
Clostridium sp. CAG:2737

Francisco Zorrilla

Chalmers University of Technology

Supervisor: Aleksej Zelezniak

Abstract

We develop a metagenomics workflow, integrating an array of existing bioinformatics and metabolic modeling tools, aimed at interrogating social interactions in bacterial communities of the human gut microbiome. From WGS metagenomic datasets, metagenome assembled genomes (MAGs) are reconstructed, which are then converted into genome-scale metabolic models (GEMs) for *in silico* simulations of cross feeding interactions within sample based communities. Abundance estimates for community members are estimated by mapping metagenomic samples to the generated MAGs, which are used in combination with the simulated cross feeding interactions for the generation of explanatory and statistically significant linear models. We conclude that there is indeed a correlation, ranging from weak to moderate, between gut microbiome members' abundance and set of metabolic cross-feeding interactions across samples. A more comprehensive analysis incorporating multiple datasets needs to be conducted to strengthen and expand the findings of this work.

1 Introduction

1.1 Historical perspective

The study of microbial communities has come a long way since Leeuwenhoek first observed microorganisms under a microscope in 1676. Since these humble origins, our understanding of microbiology, biochemistry, metabolism, and genetics has progressed at a staggering pace. Another milestone took place as the molecular structure of deoxyribonucleic acid, the molecule of heredity, was identified in 1953 [1]. Less than thirty years later, Sanger and colleagues developed the first DNA sequencing method in 1977. Sanger sequencing is reliably accurate but costly in terms of time and reagents, rendering it unfit for high throughput. A few decades later, the first commercially available high throughput (NGS) machines were released in 2005. The first whole genome shotgun (WGS) gut microbiome analysis studies were performed in 2006. Around this time, the field of metagenomics exploded in popularity as the cost of sequencing continues to drop. As sequencing becomes cheaper, greater amounts of novel and diverse genomic data becomes available for analysis. According to the NIH, the cost of sequencing a whole human genome dropped from around $95 000 000 in 2001 to around $1 000 in 2017. Today, companies such as 23andMe offer personal sequencing products and services for as little as $100 [2]. The falling cost of sequencing is visualized in Figure 1 below.

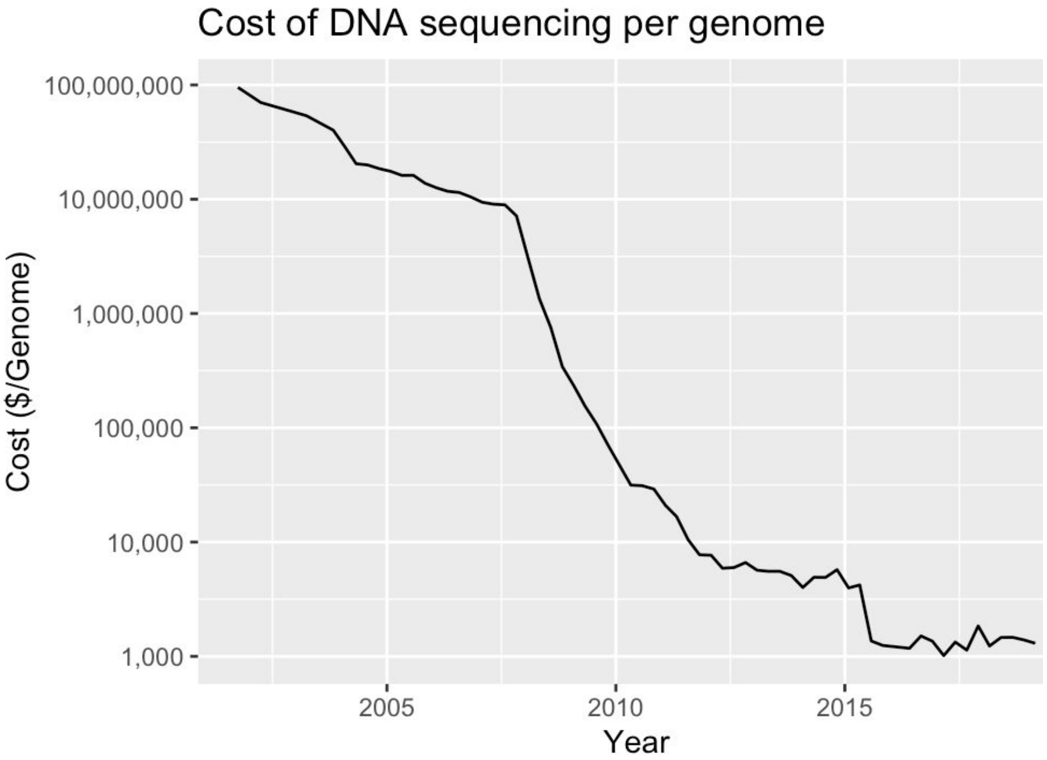


Figure 1: Decreasing cost of sequencing an entire human genome from 2001 to 2017 [3].

1.2 Metagenomics of the human gut microbiome

Nowadays, approximately two decades after the rise of NGS technologies, bioinformatic methods and tools have matured sufficiently to allow researchers to explore complex microbial communities at ever increasing levels of resolution. Presently, there is great interest in the research area of the human gut microbiome, as the list of diseases with links to gut microbiome health continues to expand. This list includes but is not limited to: IBD, atopic asthma, persistent antibiotic-induced colitis, behavioural disorders, T2DM, cardiovascular disease, autoimmune diseases, and colorectal cancer [4]. Although many of these links have been observed, mechanistic understanding of these phenomena remains to be elucidated. This is perhaps one of the reasons why much research focus has been placed on developing methods and tools for efficient and accurate species level genome recovery from WGS metagenomics data. In recent months, a number of papers aiming to explore the human gut microbiome composition and diversity have generated thousands of *de novo* metagenome assembled genomes (MAGs) of previously unknown or uncultured species [5,6,7]. A MAG is simply a text file containing assembled and species specific DNA reads from metagenomic samples. *De novo* generated MAGs are assembled without making use of reference genomes, and can therefore be used to discover new species or study species which lack a reference genome. As opposed to reference based approaches, *de novo* methods rely on the binning of reads based on nucleotide frequency, abundance, covariation in abundance across samples, and other such biological features [8]. Because *de novo* MAGs are generated directly from metagenomic samples, they provide a useful way of studying the composition and strain level variation of gut microbiome species in individuals, lending itself to the emerging personalized medicine paradigm. Notably, most of the data used in the aforementioned studies is not original, rather they take advantage of the plethora of published metagenomic datasets and databases. The *de novo* MAG generation approaches used by these papers are quite similar, and have similar to the analysis workflow of this thesis project.

1.3 Genome-scale metabolic models

MAGs can be used to generate genome scale metabolic models, also known as GEMs. These models can be seen as mathematical formulations of cellular metabolism for a given organism. In short, GEMs summarize the stoichiometry of the complete set of metabolic reactions of an organism. Using an objective function as well as other relevant biophysical and chemical constraints, solving for the feasible solution space of fluxes becomes a linear algebra problem. Notably, GEMs are quantitative and stoichiometric, as generally no kinetic information is needed for simulations. Originally GEMs were reconstructed manually, which remains a prohibitively tedious process demanding anywhere from months to years to complete [9]. Around the 2010's, tools such as RAVEN began offering semi-automated homology-based reconstruction approaches, which reduce the total reconstruction time to a number of days, but are still limited to species with available reference genomes [10]. More recently, novel top-down reconstruction-based tools have been developed, allowing reconstruction of bacterial models to complete in a matter of minutes [11]. Perhaps most significantly, these tools do not necessarily require a complete reference genome, instead they can generate models using high quality MAGs.

1.4 Aim and significance

The aim of this thesis project is to study the metabolic interactions between species within a number of gut microbiome communities. More specifically, the objective is to find statistically significant and biologically meaningful correlations between abundance and interactions. To achieve this, a workflow was developed to first generate MAGs, then convert these to GEMs, and finally perform *in silico* simulations of community-specific cross feeding interactions. Similar efforts have recently been published and can be useful resources for the scientific community [12]. While a comparable pipeline such as MetaWrap can provide a solid framework for the core tasks in a metagenomic analysis, it falls short in integrating complementary approaches such as genome scale metabolic model reconstruction or community flux balance analysis (FBA) simulations. To the researchers' best knowledge, the work presented in this thesis represents the first application of sample-specific gut microbiome community FBA simulation using MAG-based GEMs.

1.5 metaBAGpipes: A set of pipelines

The pipelines generated in this project were written in Snakemake, a python based reproducible research tool. A schematic representation of this workflow is detailed below in Figure 2. The Snakefile scripts and instructions have been made publicly available on Github (https://github.com/franciscozorrilla/metaBAGpipes).

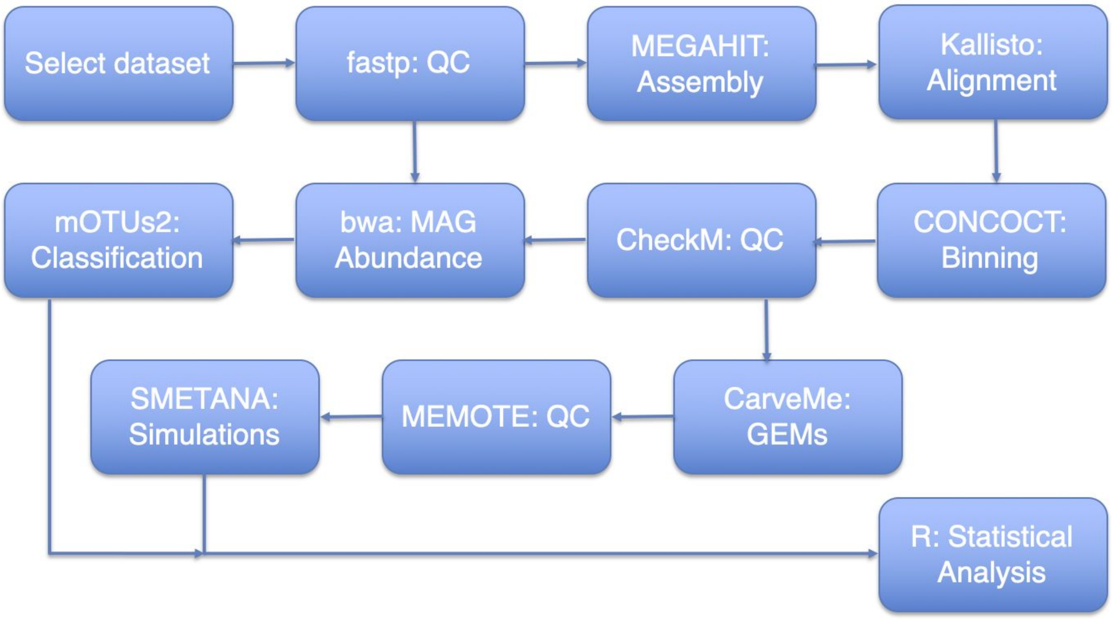


Figure 2: A schematic representation of the metaBAGpipes workflow

2 Results

2.1 Quality control of raw reads

The fastp quality control tool was used to trim low quality reads in the dataset, which contained a total of 137 sets of paired end reads (WGS). As can be seen in the upper subplots of Figure 3, on average each sample in the dataset originally consists of roughly 3 billion 100 base pair reads. Regarding the range of read counts in the dataset, some samples have as many as 6 million reads while some have as little as 1 billion reads.

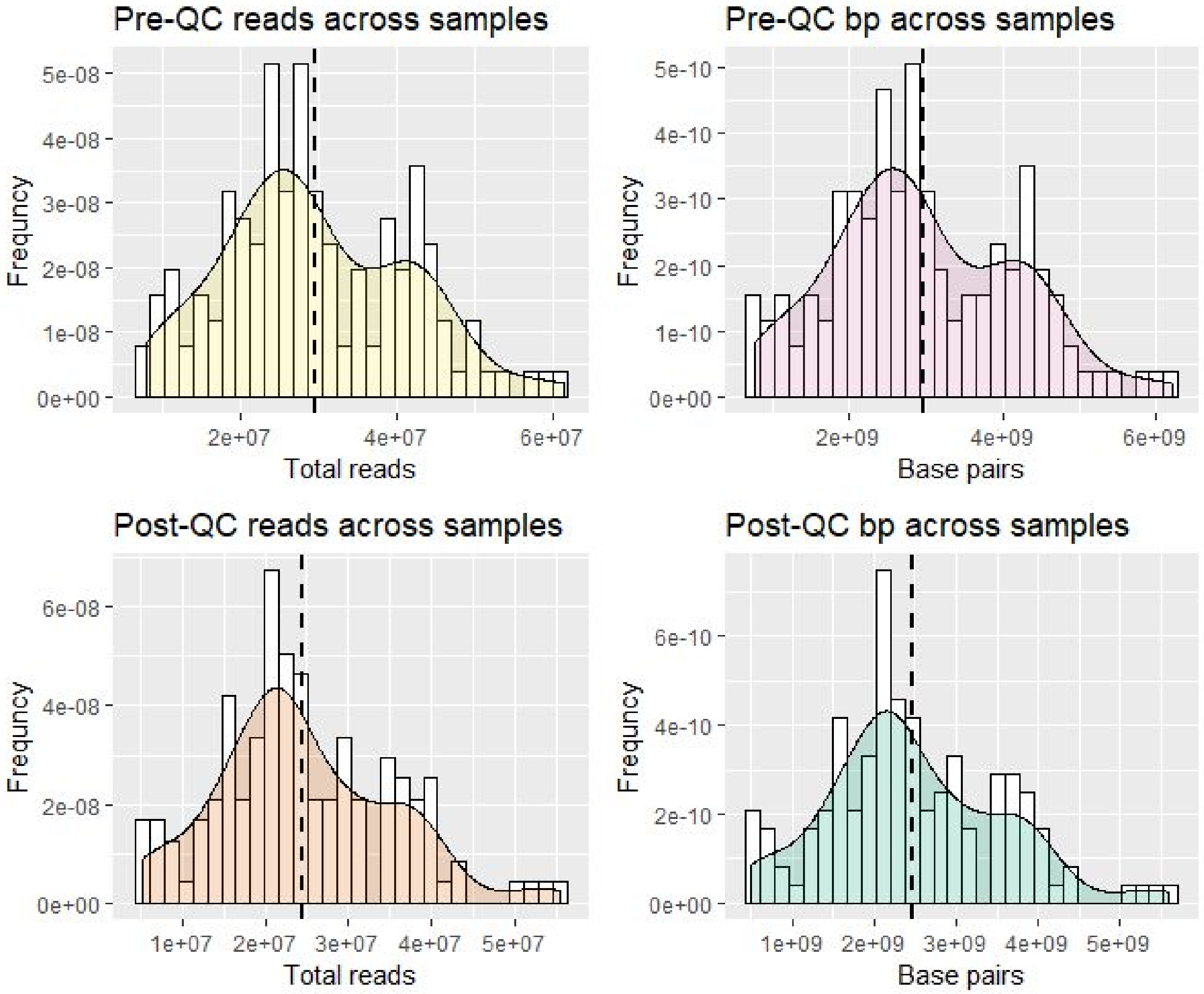


Figure 3: Histograms showing effect of quality filtering on total base pairs and reads across samples. Dashed lines show average value in each distribution.

After quality control, it appears that on average around 5 million reads are filtered out per sample, corresponding to an average loss of approximately half a billion base pairs per sample.

Based on the average values of the distributions in Figure 3, we can estimate that approximately one sixth of all information (i.e. base pair content) in dataset was filtered out due to low quality. In other words, approximately 83.33% of the total information in the dataset was kept for further processing.

2.2 Assembly into contigs

The megahit tool was used to assemble each quality filtered sample. Figure 4 shows the distribution of various key metrics corresponding to the generated per-sample assemblies. Namely, these metrics are the total number of contigs per assembly, the average contig size per assembly, the total base pair content per assembly, and the maximum contig size per assembly.

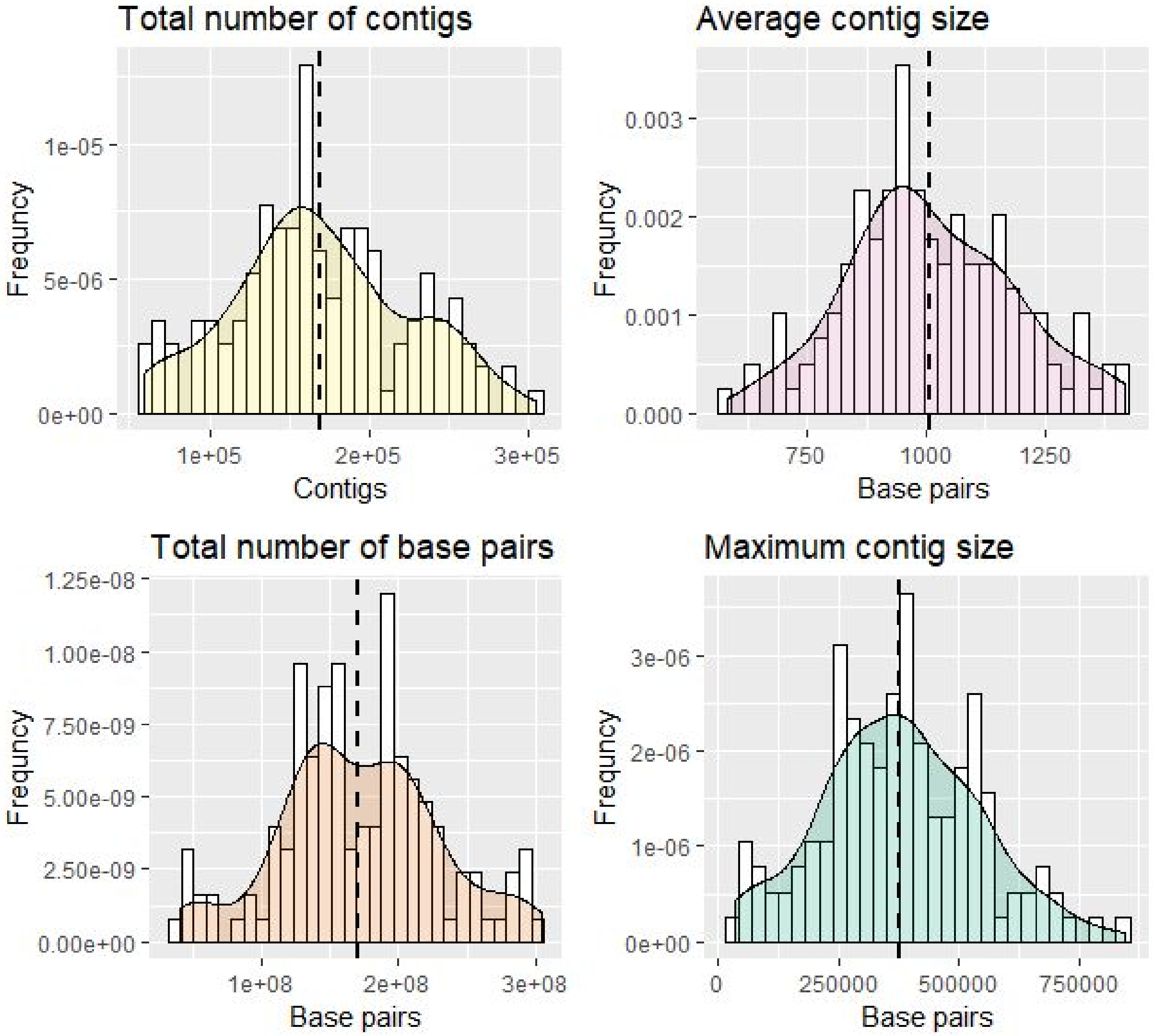


Figure 4: Histograms showing key metrics across all assemblies. Dashed lines show average value for each distribution.

As can be seen in the upper left subplot of Figure 4, the average total number of contigs across assemblies is approximately 170 000, with a range approximately between 60 000 and 300 000 contigs. The upper right subplot shows that the average contig size across assemblies is slightly above 1000 base pairs, with a range between 600 and 1400 base pairs. As can be seen in the lower left subplot, the average number of total base pairs across assemblies is approximately 171 million, with a range approximately between 40 million and 300 million base pairs. Finally, the lower right subplot shows that the average maximum contig size across assemblies is approximately 378 000 base pairs, with a range approximately between 33 000 and 846 000 base pairs.

2.3 Generation of abundance tables and binning into MAGs

The kallisto tool was used to map each assembly to each set of quality controlled (fastp) paired end reads, culminating in a total of 18 769 mapping operations which were carried out in a high performance computer cluster. The mapping files were then summarized into a total of 137 abundance tables, one for each sample. This abundance tables contains coverage information for contigs from a given sample across all other samples, which is then used to create species specific bins within each sample. The CONCOCT tool was used to organize the previously assembled contigs into species specific bins using the aforementioned abundance tables. In this research project, the term *bins* can be used interchangeably with *MAGs*, or metagenome assembled genomes. Note that these bins are also sample specific, as they were generated using only contigs from the focal sample. Since each set of MAGs corresponds to each sample’s metagenome, one can think of these sets of MAGs as each sample’s MAG-ome. In total, 41 198 bins were reconstructed across all samples in the dataset. The agglomeration of the entire dataset’s MAGs can be thought of as the dataset’s meta-MAG-ome. Statistics for these results are discussed in the following subsections.

2.4 Quality control of MAGs

The CheckM tool was used to quality filter the generated meta-MAG-ome. This is a crucial quality control step, considering that the binning algorithm uses an unsupervised approach. In order to capture as much information as possible, comparatively lax parameters were used for quality filtering. The figure below shows histograms for the distribution of total bins in each sample before and after quality control, as well as the distribution of total base pairs across bins before and after quality control.

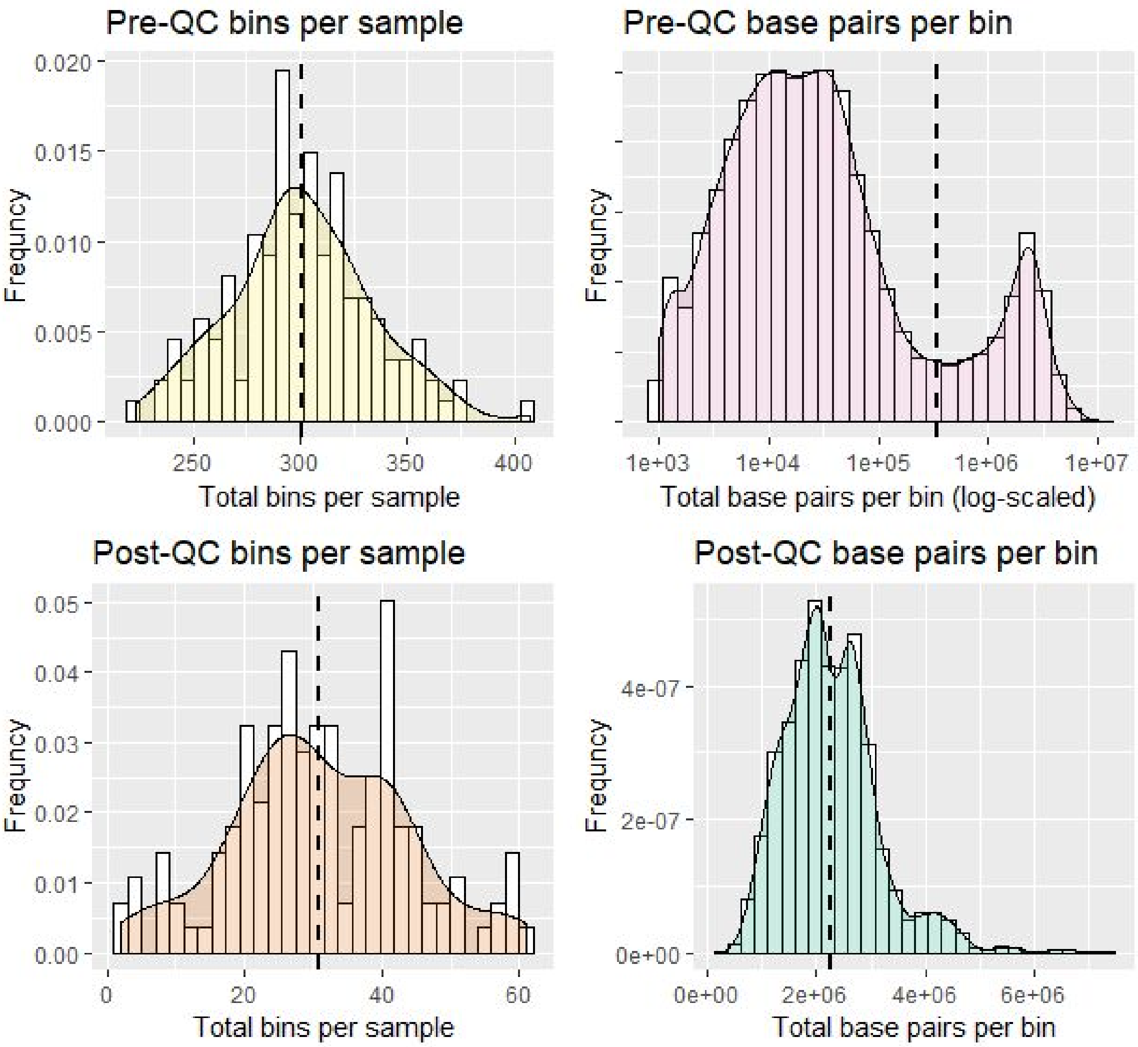


Figure 5: Histograms showing distribution of number of bins across samples before and after QC, as well as base pair content across bins before and after QC.

As can be seen in the upper left subplot of Figure 5, the pre-quality control average number of bins across samples is approximately 300, with a range roughly between 200 and 400 bins. On the other hand, the upper right subplot shows that the pre-quality control average base pair content across bins is approximately 341 000, with a large range between 1000 and 12 000 000 base pairs. This indicates that a great majority of bins contain very little genetic information. As can be seen in the lower left subplot, the post-quality control average number of bins across samples is approximately 30, with a range between 2 and 61 bins. Finally, the lower right subplot shows that the post-quality control average base pair content across bins is approximately 2 260 000, with a range between 351 000 and 7 470 000 base pairs. By taking the ratio of total base pairs across all bins after and before filtering, it was determined that 68.1% of the total bin information is retained after CheckM quality control. Note that this percentage is relative to the post-fastp-quality control total base pairs across all samples and not the total base pairs across all raw data samples. This information in visualized in the subsequent section. After quality filtering using CheckM, there were a total of 4236 MAGs across the 137 samples.

2.5 Relative abundance and classification of MAGs

MAGs were easily classified using the mOTUs2 tool. In order to generate abundance estimates for each bin (i.e. MAG), a number of mapping steps were carried out using the Burrows-Wheeler Aligner (bwa). First, each quality filtered sample was mapped to each bin that it generated, yielding the total number of reads from the focal sample mapping to each bin. Next, each sample was mapped to its corresponding MAG-ome (i.e. the per-sample concatenation of all bins), yielding the total number of reads from each sample that are represented in the corresponding MAG-ome. Taking the ratio of these two values yields the total number of reads mapped to a bin relative to the total number of reads mapped to the bin's MAG-ome. To account for the large range in genome sizes as seen in the lower right subplot of Figure 5, each ratio was then normalized by its bin size. Each ratio was then multiplied by a factor of 100 base pairs to account for the length of reads. Finally, the ratios were again multiplied by a factor of 100 in order to obtain a normalized percentage of reads corresponding to each bin relative to its MAG-ome.

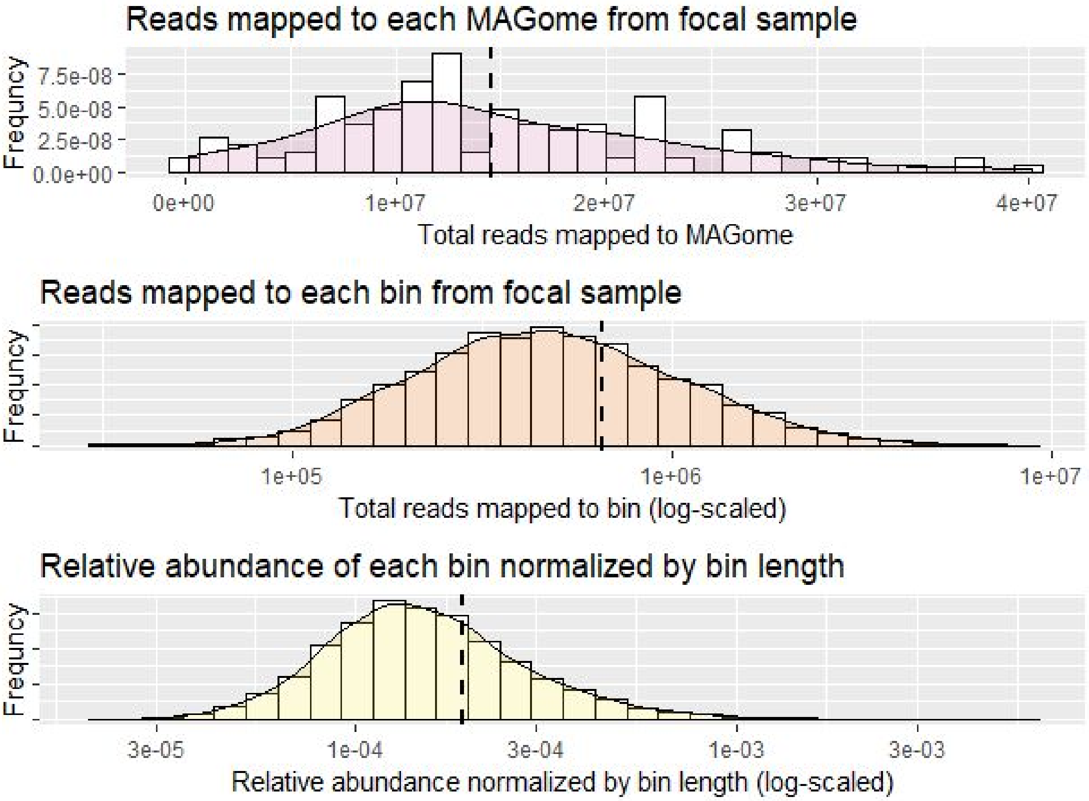


Figure 6: Histograms showing the distribution of reads mapped to each MAGome, reads mapped to each bin, and relative abundance of bins.

As can be seen in the upper subplot of Figure 6, on average there were a total of 14 500 000 reads mapped from each set of paired end quality filtered reads to its corresponding MAGome, with a range roughly between 200 000 and 40 000 000 reads. In the center subplot of Figure 6, we can see that on average there were a total of approximately 66 000 reads mapped to each bin from its parent sample (i.e. focal sample). The number of reads mapped to each bin ranges roughly between 30 000 and 7 600 000. In the bottom subplot of Figure 6, we can see that the average relative abundance normalized by bin length is approximately 1.92 e-04, with a range between 2.12 e-05 and 5.50 e-3. Using the generated mapping information, it is possible to visualize how much metagenomic information is discarded by the quality control tools (i.e. fastp and CheckM) and how much information is captured by each sample's MAGome. This was visualized using an overlay bar chart as shown in the figure below.

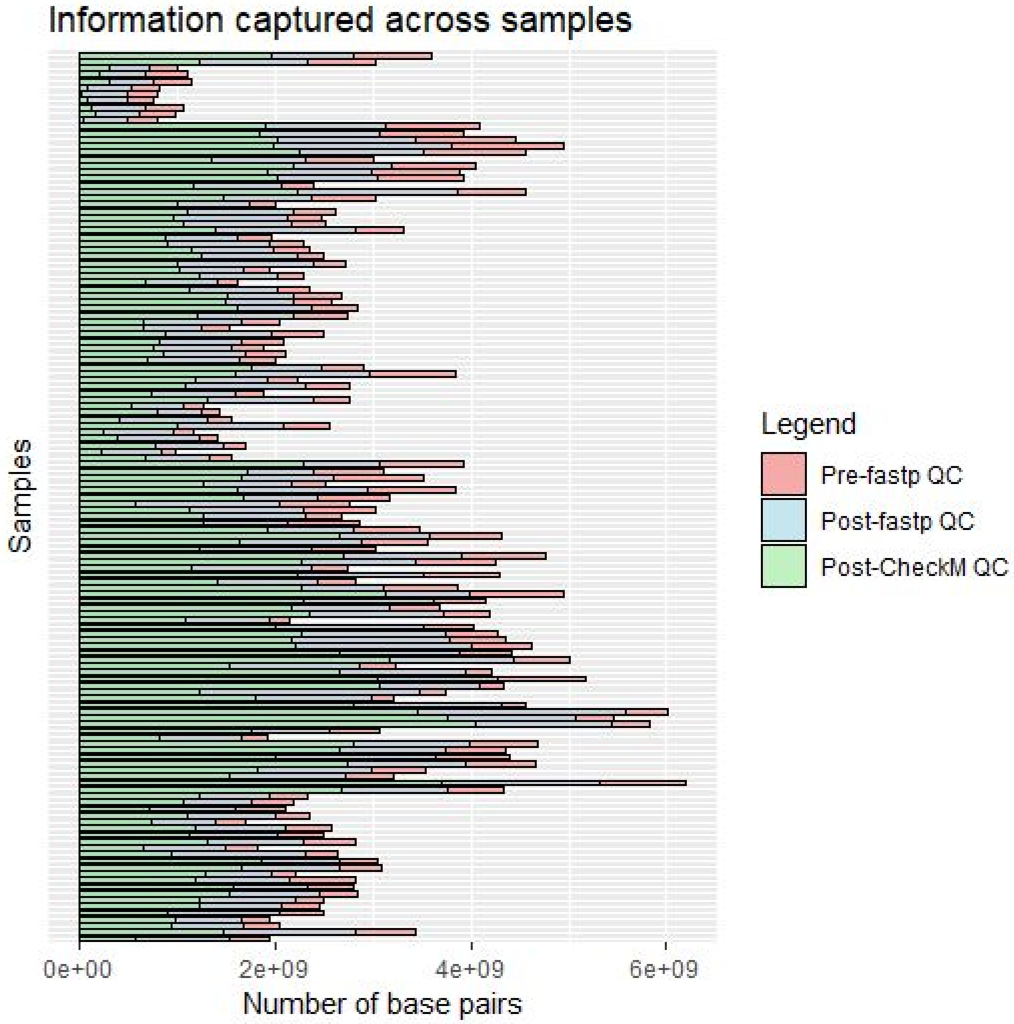


Figure 7: Overlay bar plot showing total base pairs before fastp QC, after fastp QC, and after CheckM QC.

As can be seen in Figure 7, there are a number of samples with a comparatively modest amount of information in terms of base pair content, even at the pre-fastp quality control stage of processing. This indicates that all downstream analysis, particularly the MAG generation and subsequent community simulations, may be unreliable for these “smaller” samples as the generated MAGomes may be capturing incomplete or unrealistic communities.

2.6 GEM reconstruction from MAGs

The CarveMe tool was used to automatically generate species specific genome scale metabolic models via a top-down reconstruction approach. The models were gap filled on defined gut microbiome media (dGMM) + LAB [13]. Out of the 4236 reconstructed MAGs, 96.08% were successfully used to generate species specific models, yielding a total of 4070 GEMs across the 137 samples. The distribution of unique number of metabolites per GEM, total number of reactions per GEM, and the total number of genes per GEM are summarized in the figure below.

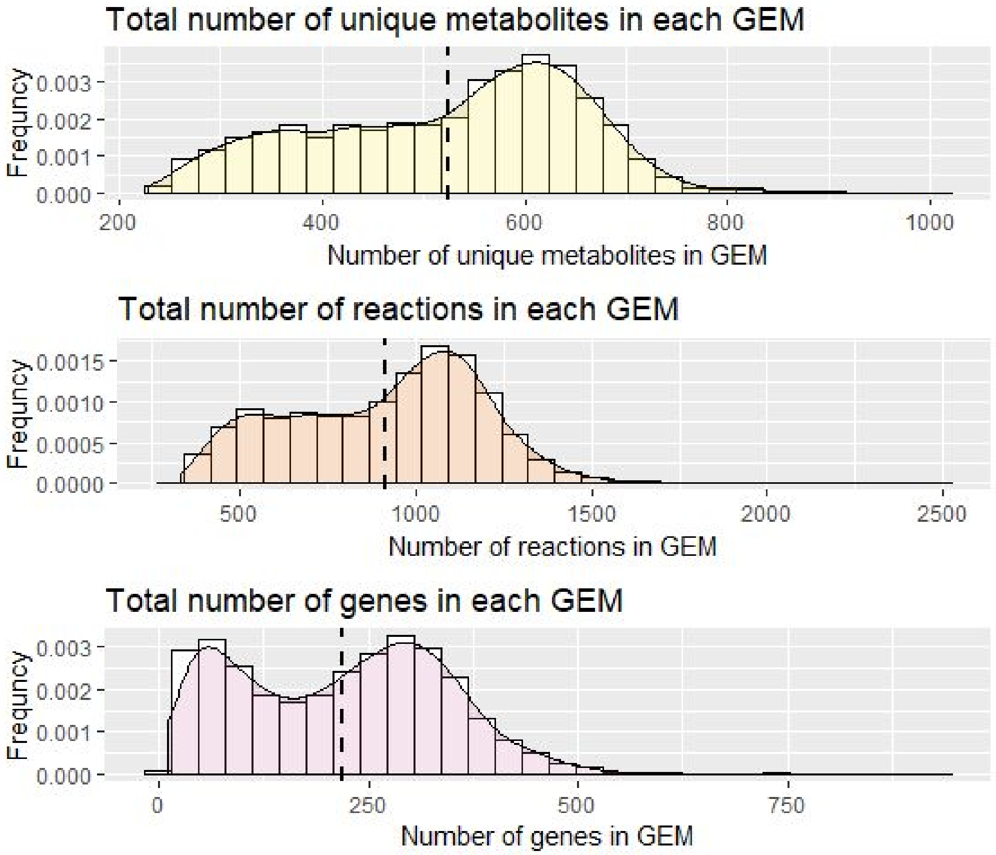


Figure 8: Distribution of unique metabolites, reactions, and genes across all generated GEMs.

As can be seen in the upper subplot of Figure 8, the average number of unique metabolites (i.e. not counting the same metabolite in different compartments) across GEMs is approximately 524, with a range between 228 and 997 unique metabolites. The center subplot shows that the average number of reactions across GEMs is around 915, with a range between 329 and 2515 reactions. The bottom subplot shows that the average number of genes across GEMs is approximately 218, with a range between 10 and 939 genes.

2.7 Quality control of GEMs

The memote tool was used to run tests on and generate reports for the reconstructed GEMs. This was primarily used during pipeline development, to ensure that the generated models were functional. No models were filtered out based on these results, considering all 4070 models were able to achieve growth.

2.8 Community metabolic simulations

The SMETANA tool was used to simulate community interactions using the MAG based GEMs. More specifically, simulations were run for each individual community of GEMs derived from each individual sample's MAG-ome. While each community simulation was run on a total of 15 different media compositions, cross feeding interactions were only observed in 7 of the media compositions. Notably, 4 out of these 7 media compositions yielded the same set of interactions. This can be explained by the fact that these 4 media compositions (M3, M4, M8, and M10) are slight variations of defined gut microbiome media and lactic acid bacteria media (dGMM + LAB). The remaining media compositions are: M5 corresponding to dGMM + LAB excluding SCFA, M7 corresponding to dGMM + LAB with only monosaccharides, and finally M11 corresponding to dGMM + LAB excluding aromatic amino acids. Each interaction was classified according to the following metabolite classes: amino acids, benzenoids, carbohydrates, inorganics, nucleosides, organic acids, complex metabolites, alcohols, and other. Below is a table summarizing the SMETANA results across all simulated communities.

Table 1: Total interaction counts across different media for different classes of metabolites

	M3/M4/M8/M10	**M5**	**M7**	**M11**
Amino acids	170	153	246	19278
Benzenoids	56	62	113	2604
Carbohydrates	155	145	645	6496
Inorganics	161	147	518	9119
Nucleosides	320	463	1181	23651
Organic acids	186	197	331	6694
Complex metabolites	213	239	526	11199
Alcohols	16	15	15	345

Other	81	239	230	4633
Total	1358	1660	3805	84019

As can be seen in Table 1, the vast majority of all observed interactions (92.5%), occur in media M11 (dGMM + LAB - Aromatic amino acids). Looking at the interaction counts across metabolite classes for each individual media, we see that the nucleoside metabolite class consistently has the highest interaction count for each media. To visualize the results shown in Table 1, the fraction of total interactions for each metabolite class was calculated for each media. These results are summarized in the radar chart below. Note that since the relative fraction is being visualized, this chart does represent the fact that there was an order of magnitude more interaction counts observed in media M11 compared to the rest of the media compositions.

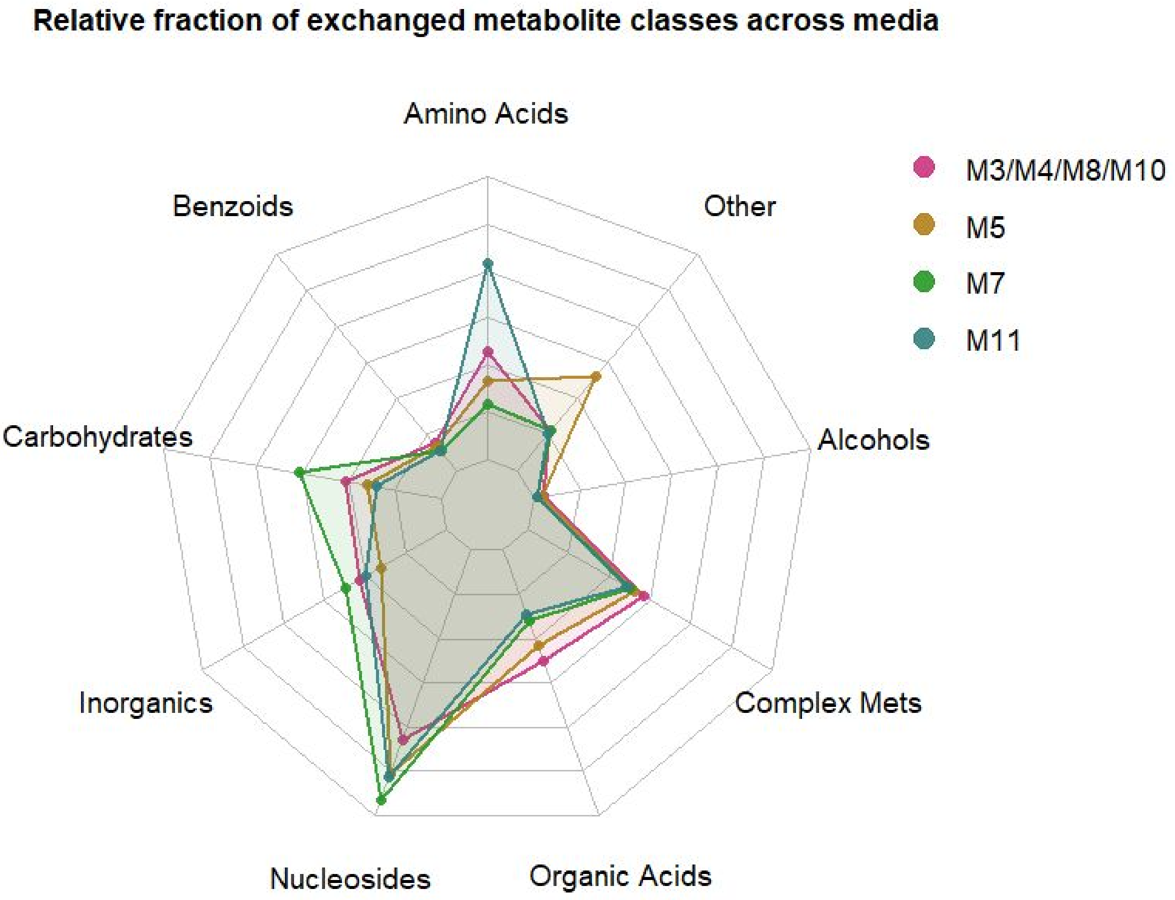


Figure 9: Radar chart showing relative fraction of interactions involving different metabolite classes across different media.

As expected, the relative fraction of amino acid exchanges were greatest in M11 (dGMM + LAB - aromatic amino acids), while the relative fraction of carbohydrates exchanges were greatest in M7 (dGMM + LAB + only monosaccharides).

2.9 Linear models

The R statistical language was used to generate multiple linear models, in an attempt to correlate the abundances of individual organisms with their set of metabolic cross feeding interactions across multiple samples. Various sets of models were produced, using various subsets of data. First, a set of models were built using metabolic interactions from all metabolite classes and all media compositions. Then, a second set of models were built using metabolic interactions from all metabolite classes, but only using interactions observed under media M11. Finally, the third set of models was built using only metabolic interactions from the nucleoside metabolite class observed in media M11. Each set of models was filtered by adjusted R^2 and adjusted p-value in order to select the top scoring models in each set. These are summarized in the tables below.

Table 2: Top linear models obtained using metabolic interactions involving all metabolite classes and all media

Taxonomy	Adjusted R^2	Samples	Adjusted p-value
[Eubacterium] eligens	0.14782	38	3.7308 e-08
unknown Eggerthella	0.11441	27	8.2791e-05
Roseburia inulinivorans	0.15521	22	2.6883e-05
Bifidobacterium longum [C]	0.11037	22	0.00029
Roseburia intestinalis	0.10357	22	0.00035
Ruminococcus sp. CAG:177	0.13269	18	2.6949 e-05
unknown Eubacterium	0.26587	15	3.7308 e-08
unknown Prevotella	0.14080	14	0.00292
Bacteroidales gen. [C Bacteroides sp./Porphyromonas sp./Parabacteroides distasonis]	0.20936	13	0.00019
Alistipes onderdonkii [C]	0.12839	13	0.00108
Eubacterium ramulus	0.13764	12	0.00574
Bacteroides stercoris	0.41756	11	5.8387 e-05

Bacteroides caccae	0.24923	11	0.00238

Having used a threshold of 0.1 to filter for the top scoring models, we can see the explanatory power of the models in Table 2 ranges between 0.1 to 0.41, while the number of samples represented by each model ranges between 11 and 38. The model for *Bacteroides stercoris* seems to have the greatest explanatory power, however the number of samples used to create the model is relatively small. The model for *Eubacterium eligens* captures the greatest number of samples, however it has a modest explanatory power.

Table 3: Top linear models using metabolic interactions involving all metabolite classes and only M11 media.

Taxonomy	Adjusted R^2	Samples	Adjusted p-value
[Eubacterium] hallii	0.10889	66	5.5570 e-07
unknown Eggerthella	0.10127	26	0.00877
Roseburia intestinalis	0.11325	22	0.00496
[Eubacterium] siraeum	0.11768	15	0.02743
Clostridium sp. CAG:343	0.11240	14	0.04115
Bacteroidales gen. [C Bacteroides sp./Porphyromonas sp./Parabacteroides distasonis]	0.17172	13	0.00727
Eubacterium ramulus	0.35924	12	3.6221 e-05
Alistipes onderdonkii [C]	0.21989	12	0.00189
Bacteroides caccae	0.24733	11	0.00496
Bacteroides stercoris	0.48145	9	0.00010

Table 3 more clearly shows how the explanatory power of models appears to be negatively correlated to the number of samples captured by the model. By narrowing down the dataset used to build the models we can see that the explanatory power of some models increases relative to Table 2. For example, we can see that the model for *Bacteroides stercoris* appears again, this time with less samples and greater explanatory power relative to the model for *Bacteroides stercoris* present in Table 2. This pattern can also be observed for the models of *Eubacterium ramulus* and *Alistipes onderdonkii*.

Table 4: Top linear models using metabolic interactions involving only nucleosides and only in M11 media

Taxonomy	Adjusted R^2	Samples	Bins	Adjusted p-value
unknown Clostridium	0.14514	62	91	0.03111
[Eubacterium] hallii	0.25212	58	58	0.02512

Table 4 yields a notably lesser number of significant models compared to Table 2 or Table 3, which can be explained by the fact that we have only used a fraction of the total data (approximately 28% of the interactions observed in media M11).

3 Discussion

3.1 MAGs

While the average number of post-quality control bins per sample may be less than expected for human gut microbiome communities, it is consistent with results obtained in similar research efforts. In fact, the average number of MAGs per sample obtained from this workflow is between two to four times greater than what has been reported in similar state of the art papers [5,6,7]. Indeed, this could be partly explained by differences in the strictness of CheckM parameters used for quality filtering of MAGs. Furthermore, due to limitations in the depth of sequencing in the samples, it is quite likely that the generated and quality filtered MAGs correspond primarily to organisms present at high abundances. Unexpectedly, there appears to be little to no correlation between the number of MAGs obtained per sample before and after quality filtering, as shown in the figure below.

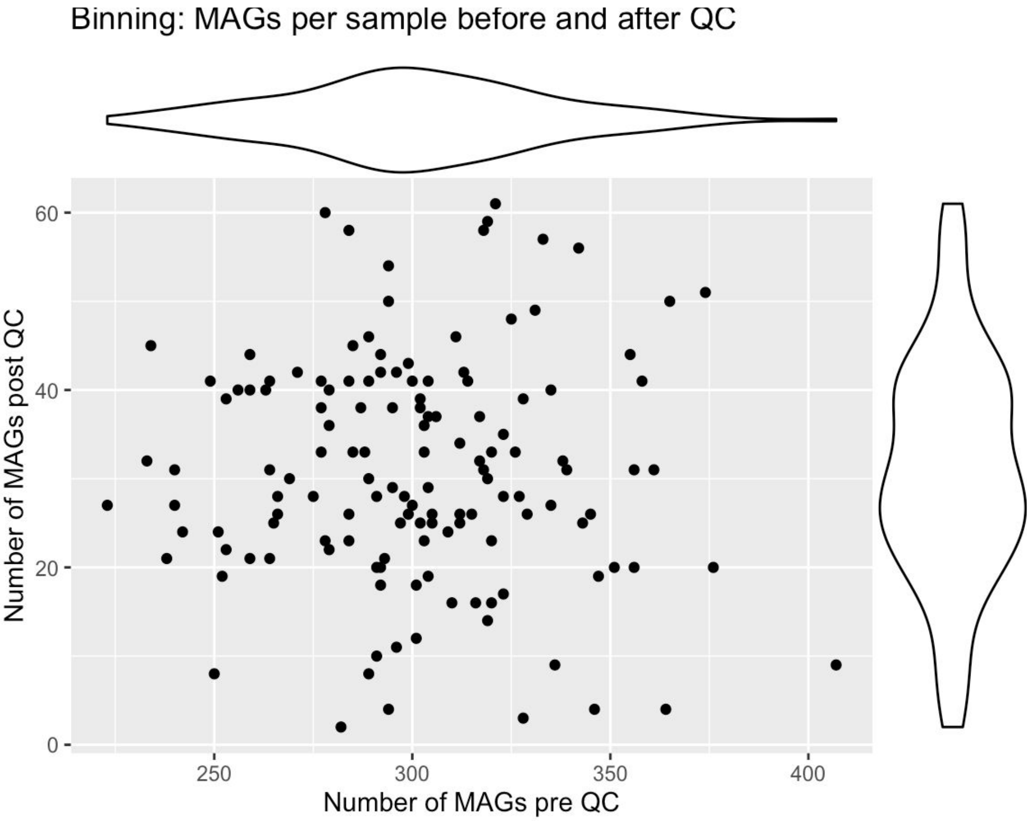


Figure 10: Scatter with marginal violin plots showing number of MAGs per sample before and after quality filtering.

As can be seen in the figure above, the number of pre-quality control MAGs appears to be a poor indicator of the number of post-quality control MAGs, and vice versa. This could likely be due to the differences in content and quality across samples, as highlighted by the overlay bar graph in Figure 7. Figure 10 also underscores the tradeoff between capturing as much information as possible and the quality of resulting MAGs, i.e. obtaining many MAGs with varying quality vs obtaining few MAGs with high quality. This indicates that additional re-binning steps may be necessary to avoid the discarding of useful information present in the lower quality MAGs.

Additionally, it was noted that a large number of MAGs contain highly fragmented genomes as shown in the figure below. This suggests that an additional assembly step may be necessary within each generated bin, so as to increase the quality and contiguity of generated MAGs.

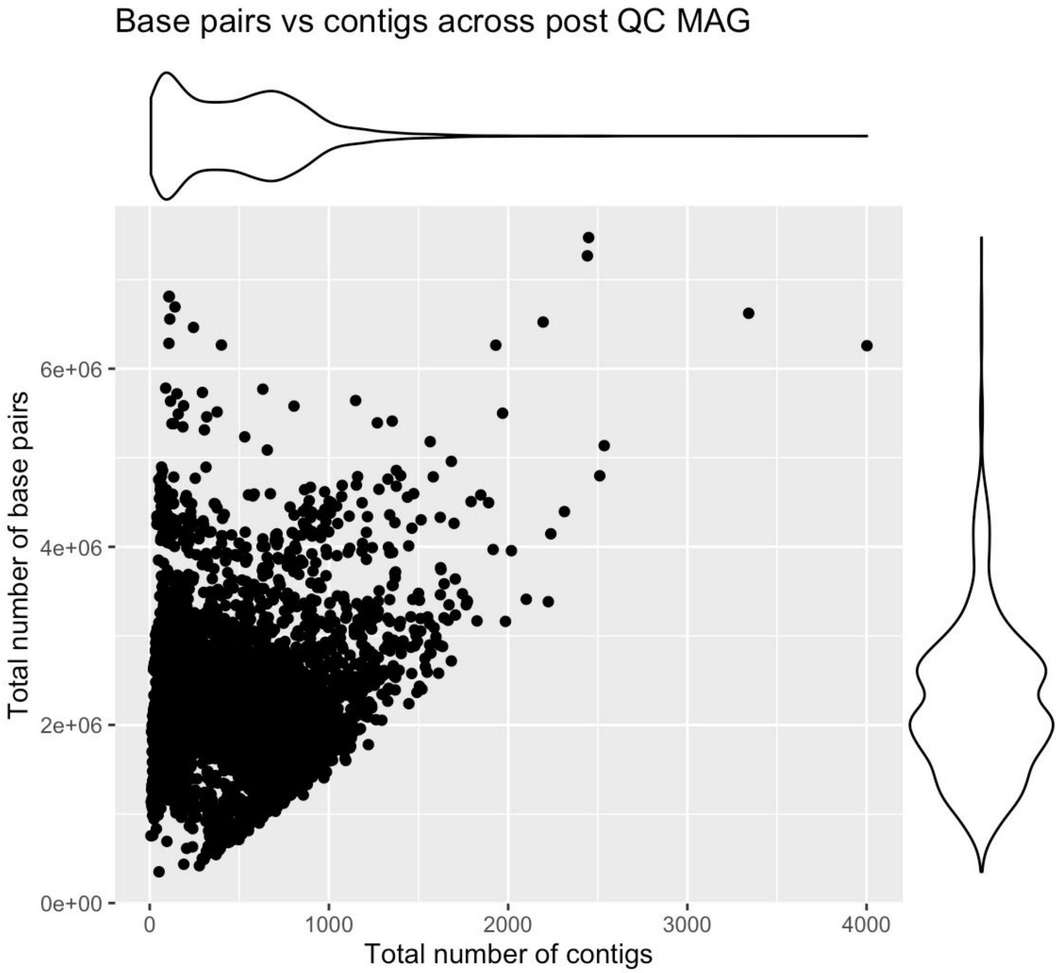


Figure 11: Scatter with marginal violin plots showing the total base

Finally, it should be noted that the dataset used in this workflow was taken from a study which attempted to characterize the gut microbiome of patients based on fecal samples [14]. However, a recent study which invasively characterized the local mucosal and lumen microbiome of different biogeographical regions of the human gastrointestinal tract found that there is only partial correlation with stool samples [15]. In other words, communities of MAGs reconstructed from fecal samples may not necessarily be representative of real gut microbiome communities. In fact, some MAGs may even be amalgamations of strains from the same species, but from different biogeographical regions of the human gastrointestinal tract. These limitations likely affect the quality of results obtained, and it would be an interesting future step to use WGS data directly from the different biogeographical regions of the gastrointestinal tract instead of relying on fecal samples.

3.2 GEMs

Perhaps most strikingly, the average number of metabolic genes across reconstructed GEMs is quite low. This phenomenon could be explained by a number of reasons: firstly, the “completeness” parameter used for the filtering of MAGs was quite lax (minimum 40% completeness required to pass). Additionally, as shown in the figure below as well as the figure above, a significant portion of MAGs contain highly fragmented genomes. In fact, the mean number of contigs per bin is 499, while the median sits at 478 contigs per bin. As previously mentioned, an additional assembly step will be necessary to increase the contiguity of MAGs. This would likely allow CarveMe to identify previously missed metabolic genes, which could end up being spread across multiple contigs without the additional assembly step. However, it is important to note that there are likely many genes captured in the MAGs which are not translated into the GEMs. This has to do with the methodology of CarveMe, which relies on a set of around 70 manually curated models that mostly correspond to various strains of *E. coli*. That is, even though the generated MAGs may correspond to novel or strain specific organisms, unless the genes present in these MAGs are also present in the aforementioned repository of GEMs, they will not be present in the generated GEMs.

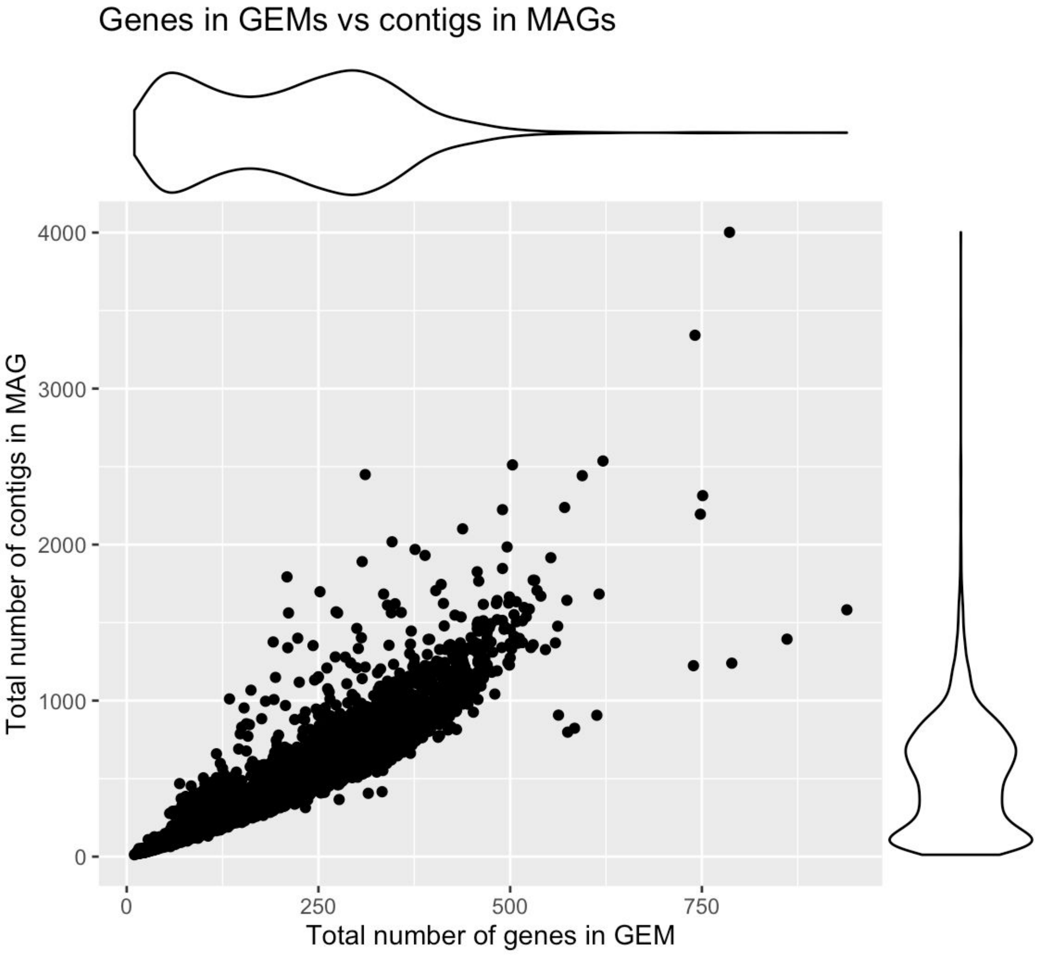


Figure 12: Scatter with marginal violin plots showing genes across GEMs against contigs across MAGs

3.3 Linear models

Based on a comparison of the generated linear models presented in Table 2 and Table 3, it appears that narrowing down the dataset to individual metabolite classes can yield models with greater statistical significance, particularly regarding presence across a greater number of samples. However, this narrowing down of the dataset comes at the cost of a reduced list of significant models. Notably, the explanatory power of the linear models shown in Table 2 and Table 3 is quite modest, never exceeding an R^2 value of 0.5. While these results may appear underwhelming, it could very likely be the case that there are other significant determinants of abundance besides metabolic cross feeding interactions. For example, many of the gut microbiome members' abundance could be influenced by immune system protein interactions, diet, lifestyle, medical conditions, etc. To attempt to control for some of these influencing factors, it would be interesting to carry out this workflow on a dataset consisting of samples from a single individual, perhaps across a number of different points in the gastrointestinal tract, or across a determined period of time. Furthermore, these samples should be sequenced to the greatest possible depth, in order to facilitate the reconstruction of MAGs corresponding to species of low abundance. In conclusion, the obtained results suggest that there is a shortage of data, specifically regarding the number of samples used in the analysis. Additionally, it is important to mention that the dataset consists of elderly european women with normal, impaired, or diabetic glucose control, potentially adding another layer of confounding effects. This is noteworthy considering that these conditions are likely to significantly influence the abundance and/or cross-feeding interactions within the patient's gut microbiome. To address this limitation, a larger number of datasets should be analyzed with the workflow carried out in this report. Additionally, we note that the top scoring model in Table 4 has a greater number of bins than samples. This indicates a high degree of strain heterogeneity or an inadequate binning and/or taxonomic classification performance. Considering that the assigned taxonomy is "unknown Clostridium", it is possible that a combination of the suggested explanations is true.

Finally, it should also be mentioned that there are limitations with the community modeling methodology of SMETANA. In short, when simulating cross feeding interactions, SMETANA assumes an equal growth rate of 1 hr^{-1} for all community members. While this is a useful and perhaps necessary simplification in the absence of *a priori* knowledge of growth rate, it is also likely adding some degree of noise and error to the results. This limitation could be overcome by estimating growth rate values for each community member, using a tool such as GRiD [16], which could then be used as inputs to SMETANA.

4 Methods

For most tasks in a given metagenomics workflow there exist a number of different tools to choose from. In this project, tool choice was influenced by user friendliness, community suggestions, and publication date/journal. Below is a table summarizing the open source tools employed in this thesis.

Table 5: Summary of tools used in workflow

Tool Name	Task	Publisher, Year	GitHub
fastp	Raw read quality control	Bioinformatics, 2018 [17]	https://github.com/OpenGene/fastp
MEGAHIT	Raw read assembly	Bioinformatics, 2015 [18]	https://github.com/voutcn/megahit
Kallisto	Read abundance quantification	Nature Biotechnology, 2016 [19]	https://github.com/pachterlab/kallisto
CONCOCT	Contig binning into MAGs	Nature Methods, 2014 [8]	https://github.com/BinPro/CONCOCT
CheckM	MAG quality control	Genome Research, 2015 [20]	https://github.com/Ecogenomics/CheckM
bwa	MAG abundance estimation	Bioinformatics, 2009 [21]	https://github.com/lh3/bwa
mOTUs2	MAG classification	Nature, 2019 [22]	https://github.com/motu-tool/mOTUs_v2
CarveMe	GEM reconstruction	Nucleic Acids Research, 2018 [11]	https://github.com/cdanielmachado/carveme
memote	GEM quality control	bioRxiv, 2018 [23]	https://github.com/opencobra/memote
SMETANA	Community FBA simulations	PNAS, 2015 [24]	https://github.com/cdanielmachado/smetana

All jobs were run on the Vera C3SE high performance computer cluster (HPC) available at Chalmers University of Technology. The tools listed in Table 5 were installed in one of two Anaconda3 environments, one for tools requiring Python > 3.0 and another for tools requiring Python < 3.0. The developed workflow was compiled as a set of Snakefiles, which are hosted on GitHub.

4.1 Quality control of raw reads

Each set of paired end reads was quality filtered using the default fastp parameters, and each job was run on 8 cores. Result statistics were generated as json and html files.

```
rule qfilter:
	input:
	R1=config["paths"]["raw_reads"]+"/{ref}_1.fastq.gz",
	R2=config["paths"]["raw_reads"]+"/{ref}_2.fastq.gz"
	output:
	R1="qfiltered/{ref}/{ref}_1.fastq.gz",
	R2="qfiltered/{ref}/{ref}_2.fastq.gz"
	shell:
	"""
	fastp --thread 8 -i {input.R1} -I {input.R2} -o {output.R1} -O
{output.R2} -j $(dirname {output.R1})/$(echo $( basename $(dirname
{output.R1}))).json -h $(dirname {output.R1})/$(echo $( basename
$(dirname {output.R1}))).html
	"""
```

4.2 Assembly into contigs

Assemblies were generated for each set of paired end reads using 32 cores, using the meta-large parameter of the megahit tool.

```
rule megahit:
	input:
	R1="qfiltered/{ref}/{ref}_1.fastq.gz",
	R2="qfiltered/{ref}/{ref}_2.fastq.gz"
	output:
	"megahitAssembly/{ref}/final.contigs.fa"
	shell:
	"""
	rm -r $(dirname {output})
	megahit -t {config[megahit_params][threads]} --tmp-dir $TMPDIR
--presets meta-large --verbose -1 {input.R1} -2 {input.R2} -o
{config[paths][concoct_run]}/$(dirname {output})
	rm -r $(dirname {output})/intermediate_contigs
	"""
```

The resulting assemblies were then cut into 10 kilobase chunks using the CONCOCT cut_up_fasta python script, as suggested by the CONCOCT workflow.

```
rule cutcontigs:
	input:
	"megahitAssembly/{ref}/final.contigs.fa"
	output:
      "contigs/{ref}/megahit_c10K.fa"
      shell:
      """
      set +u;source activate concoct_env;set -u;
      cd $(dirname {input})
      python {config[cutcontigs_params][script_dir]} -c 10000 -o 0 -m
$(basename {input}) > $(basename {output});
      mv $(basename {output}) {config[paths][concoct_run]}/$(dirname
{output});
      """
```

4.3 Read abundance quantification

An index was constructed for each assembly using the default parameters of the kallisto index command.

```
rule kallistoBuild:
      input:
      "contigs/{ref}/megahit_c10K.fa"
      output:
      "quantification/kallistoIndices/{ref}.kaix"
      shell:
      """
      set +u;source activate checkm_env;set -u;
      kallisto index {input} -i {output}
      """
```

Each set of quality filtered reads was mapped to each assembly index in order to generate contig abundance tables across samples. The kallisto quant command was used with default parameters and 32 cores.

```
rule kallistoQuant:
      input:
R1=config["paths"]["concoct_run"]+"/qfiltered/{names}/{names}_1.fastq
.gz",
R2=config["paths"]["concoct_run"]+"/qfiltered/{names}/{names}_2.fastq
.gz",
      index="quantification/kallistoIndices/{ref}.kaix",
      indexDummy="quantification/kallistoIndices/indexDummy.txt",
      output:
      "quantification/ref_{ref}/{names}/abundance.tsv.gz"
      shell:
      """
```

```
     set +u;source activate checkm_env;set -u;
     cd $TMPDIR
     kallisto quant --threads {config[kallisto_params][threads]}
--plaintext -i {config[paths][concoct_run]}/{input.index} -o $TMPDIR
{input.R1} {input.R2}
     gzip abundance.tsv
     cp $(basename {output}) {config[paths][concoct_run]}/$(dirname
{output})
     cp run_info.json {config[paths][concoct_run]}/$(dirname
{output})
     rm $(basename {output})
     rm run_info.json
     cd {config[paths][concoct_run]}
     """
```

Abundance results were summarized into a single file for each sample using the CONCOCT kallisto2concoctTable python script.

```
rule kallisto2concoctTable:
     input:
     abundances = lambda wildcards:
expand("quantification/ref_{ref}/{{names}}/abundance.tsv.gz".format(r
ef=wildcards.concoct), names = names),
     abdunaceDummy= "quantification/quantDummy.txt"
     output:
     "concoct_input/{concoct}_concoct_inputtableR.tsv"
     params:
     files= names
     shell:
     """
     set +u;source activate concoct_env;set -u;
     python {config[kallisto_params][script]} \
           --samplenames <(for s in {params.files}; do echo $s; done)
\
                {input.abundances} > {output}
     """
```

4.4 Binning into MAGs

CONCOCT was run for each sample using 32 cores and a cluster parameter value of 1000.

```
rule concoct:
     input:
```

```
     table="concoct_input/{names}_concoct_inputtableR.tsv",
     comp="contigs/{names}/megahit_c10K.fa"
     output:
     "concoct_output/{names}/clustering_gt1000.csv"
     shell:
     """
     set +u;source activate concoct_env;set -u;
     cd {config[paths][concoct_run]};
     concoct --coverage_file {input.table} --composition_file
{input.comp} -b $(dirname {output}) -t
{config[concoct_params][threads]} -c
{config[concoct_params][clusters]}
     """
```

As suggested by the CONCOCT workflow, the contigs were re-merged into their assembled length using the merge_cutup_clustering python script.

```
rule mergeClustering:
     input:
     "concoct_output/{names}/clustering_gt1000.csv"
     output:
     "concoct_output/{names}/clustering_merged.csv"
     shell:
     """
     set +u;source activate concoct_env;set -u;
     merge_cutup_clustering.py {input} > {output}
     """
```

Bins were then extracted from assemblies using the CONCOCT extract_fasta_bins python script.

```
rule extractBins:
     input:
     clustering="concoct_output/{names}/clustering_merged.csv",
     OGcontigs="megahitAssembly/{names}/final.contigs.fa"
     output:
     "bins/{names}"
     shell:
     """
     set +u;source activate concoct_env;set -u;
     mkdir -p {output}
     extract_fasta_bins.py {input.OGcontigs} {input.clustering}
--output_path {output}
```

```
      """
```

4.5 Quality control of MAGs

The CheckM lineage_wf command was used to identify marker genes in bins with default parameters and 48 cores.

```
rule checkM:
      input:
      "bins/{names}"
      output:
      "checkm_out/{names}/lineage.ms"
      shell:
      """
      set +u;source activate checkm_env;set -u;
      checkm lineage_wf --tmpdir $TMPDIR -x fa -t
{config[checkM_params][threads]} --pplacer_threads
{config[checkM_params][threads]} {input} $(dirname {output})
      """
```

The CheckM qa command was used to output tab delimited tables containing contamination and completeness estimates for MAGs.

```
rule checkMtable:
      input:
      "checkm_out/{names}/lineage.ms"
      output:
      "checkm_out/{names}/binTable.txt"
      shell:
      """
      set +u;source activate checkm_env;set -u;
      checkm qa --tmpdir $TMPDIR --tab_table $(dirname
{output})/lineage.ms $(dirname {output}) > {output}
      """
```

Bins were filtered according to contamination and completeness cutoffs. More specifically bins with less than 40% completeness or more than 40% contamination were discarded.

```
rule binFilter:
      input:
      bins= "bins/{names}",
      table= "checkm_out/{names}/binTable.txt"
      output:
```

```
      "bins_filtered/{names}"
      shell:
      """
      mkdir -p {output}
      {config[checkM_params][filterScript]} {input.bins}
{input.table} {output} --min_completeness
{config[checkM_params][comp]} --max_contamination
{config[checkM_params][cont]}
      """
```

4.6 Abundance and classification of MAGs

MAGs were taxonomically classified based on marker genes using a mOTUs2 based script (https://github.com/AlessioMilanese/classify-genomes). Each job was allocated 16 cores.

```
rule classifyMotus2:
      input:
bins="/c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe/bins_organ
ized/{ref}",
      script="/c3se/users/zorrilla/Vera/temp/classify-genomes"
      output:
      "classify-motus2/{ref}"
      shell:
      """
      set +u;source activate concoct_env;set -u
      cd {config[paths][concoct_run]}
      mkdir -p classify-motus2/$(basename {output})
      cd $TMPDIR
      cp -r {input.bins}/* .
      cp -r {input.script}/* .
      for bin in ERR*.fa; do
            echo "RUNNING BIN $bin"
            $PWD/classify-genomes $bin -t 16 -o $(echo $bin|sed
's/.fa/.taxonomy/')
            echo "DONE"
            echo " "
            cp *.taxonomy
/c3se/NOBACKUP/users/zorrilla/binning/perSamplePipe/classify-motus2/$
(basename {output})
            rm *.taxonomy
            rm $bin
      done
      cd {config[paths][concoct_run]}
      """
```

The relative abundance for each bin *i* was estimated according to the following expression:

$$relativeAbundance_i = \frac{x_i \cdot L}{y_i \cdot z} \cdot 100 \qquad \text{(Equation 1)}$$

Where x_i represents the number of reads mapped to bin *i* from its corresponding set of paired end reads, *L* represents the length of each read (i.e. 100 bp), y_i represents the base pair length of bin *i*, and *z* represents the total number of reads mapped to the MAGome of the sample (i.e. concatenation of all bins from sample). Finally, the resulting value is multiplied by 100 to obtain a normalized percentage relative abundance.

The rule binCoverage2 was used to calculate values of *z*. An index was first created for a sample's MAGome, which is then used to map reads from the corresponding quality filtered set of paired end reads using 16 cores. The resulting SAM file is then converted to BAM format and sorted. The sorted file is then used as an input to the samtools flagstat command to generate a report file containing the value of *z*.

```
rule binCoverage2:
      input:
      bins =
"/c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe/bins_organized/
",
      qfiltered =
"/c3se/users/zorrilla/Vera/binningBackup/perSamplePipe/qfiltered/{ref
}"
      output:
      "binCoverage2/{ref}.fa/map.stats"
      shell:
      """
      cd {config[paths][concoct_run]}
      mkdir -p binCoverage2
      cd binCoverage2
      mkdir -p $(basename ${input.qfiltered})
      cd $TMPDIR
      cp {input.bins}$(basename {input.qfiltered})/* .
      cat ERR*.fa > $(basename {input.qfiltered}).fa
      cp {input.qfiltered}/*.gz .
      bwa index $(basename {input.qfiltered}).fa
      bwa mem -t 16 $(basename {input.qfiltered}).fa $(echo
*_1.fastq.gz) $(echo *_2.fastq.gz) > $(basename
{input.qfiltered}).sam
```

```
      samtools view -@ 16 -Sb $(basename {input.qfiltered}).sam >
$(basename {input.qfiltered}).bam
      samtools sort -@ 16 $(basename {input.qfiltered}).bam >
$(basename {input.qfiltered}).sort
      samtools flagstat $(basename {input.qfiltered}).sort >
map.stats
      cp map.stats
{config[paths][concoct_run]}/binCoverage2/$(basename
{input.qfiltered})
      cd {config[paths][concoct_run]}
      """
```

The rule binCoverage3 below is used to calculate the values of x_i and y_i, mirroring the procedure described for the above described binCoverage2 rule.

```
rule binCoverage3:
      input:
      bins =
"/c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe/bins_organized/
{ref}",
      qfiltered =
"/c3se/users/zorrilla/Vera/binningBackup/perSamplePipe/qfiltered/{ref
}"
      output:
      "binCoverage3/{ref}"
      shell:
      """
      cd {config[paths][concoct_run]}
      mkdir -p binCoverage3
      cd binCoverage3
      mkdir -p $(basename ${input.qfiltered})
      cd $TMPDIR
      cp {input.bins}/* {input.qfiltered}/*.gz .
      for bin in ERR*.fa;do
            mkdir -p $(echo "$bin"| sed "s/.fa//")
            cp  $bin $(echo "$bin"| sed "s/.fa//")
            cd $(echo "$bin"| sed "s/.fa//")
            bwa index $bin
            bwa mem -t 16 $bin $(echo "$TMPDIR/$bin"| sed
"s/_.*$/_1.fastq.gz/") $(echo "$TMPDIR/$bin"| sed
"s/_.*$/_2.fastq.gz/") > $(echo "$bin"|sed "s/.fa/.sam/")
            samtools view -@ 16 -Sb $(echo "$bin"|sed "s/.fa/.sam/") >
$(echo "$bin"|sed "s/.fa/.bam/")
```

```
           samtools sort -@ 16 $(echo "$bin"|sed "s/.fa/.bam/") >
$(echo "$bin"|sed "s/.fa/.sort/")
           samtools flagstat $(echo "$bin"|sed "s/.fa/.sort/") >
$(echo "$bin"|sed "s/.fa/.map/")
           echo -n "Bin Length = " >> $(echo "$bin"|sed
"s/.fa/.map/")
           less $bin|grep len| awk -F' ' '{{print $NF}}'|sed
's/len=//'|awk '{{sum+=$NF;}}END{{print sum;}}' >> $(echo "$bin"|sed
"s/.fa/.map/")
           cp *.map
{config[paths][concoct_run]}/binCoverage3/$(basename
${input.qfiltered})
           cd ..
           rm -r $(echo "$bin"| sed "s/.fa//")
      done
      """
```

Finally, the rule binFraction below computes the relative abundance of each bin *i* as described by Equation 1.

```
rule binFraction:
      shell:
      """
      cd
/c3se/NOBACKUP/users/zorrilla/binning/perSamplePipe/binCoverage3/
      for sample in ERR*;do
           cd $sample
           for bin in ERR*.map;do
                echo -n "$bin" >> binAbundance.fraction
                echo -n $'\t' >> binAbundance.fraction
                X=$(less $bin|grep "mapped ("|awk -F' ' '{print $1}')
                Y=$(less $bin|tail -n 1|awk -F' ' '{print $4}')
                Z=$(less
"/c3se/NOBACKUP/users/zorrilla/binning/perSamplePipe/binCoverage2/$sa
mple/map.stats"|grep "mapped ("|awk -F' ' '{print $1}')
                awk -v x="$X" -v y="$Y" -v z="$Z" 'BEGIN{print
(x/y/z) * 100}' >> binAbundance.fraction
           done
           cd ..
      done
      """
```

4.7 GEM reconstructions from MAGs

GEMs were generated and gap-filled using the M3 media, corresponding to defined gut microbiome media + lactic acid bacteria media [13]. Additionally, the models were generated in fbc2 format. Eight cores were allocated to each job.

```
rule carveme:
      input:
      "bins_organized/{fasta}.fa"
      output:
      "carvemeOut-GFwM3/{fasta}.xml"
      shell:
      """
      set +u;source activate concoct_env;set -u
      cp {input} $TMPDIR
      cd $TMPDIR
      carve -g M3 -v --fbc2 --dna $(basename {input}) -o $(basename
{output})
      cp $(basename {output})
/c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe/carvemeOut-GFwM3
      cd /c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe
      """
```

4.8 Quality control of GEMs

The memote report snapshot and memote run commands were used to generate model statistics.

```
rule memote:
      input: "carvemeOut/{fasta}.xml"
      output: "carvemeOut/{fasta}.html"
      shell:
      """
      set +u;source activate concoct_env;set -u
      memote report snapshot --filename $(echo {input}|sed
's/.xml/.html/') {input} #generate .html report
      memote run {input}> {input}_summary.txt #generate quick
printout of model summary
      """
```

4.9 Community metabolic simulations

The SMETANA --detailed flag was used to simulate cross feeding interactions within each sample-based community across a number of different media [13]. The CPLEX solver was used, and each job was allocated 64 cores.

```
rule smetana:
      input:
"/c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe/SampleNames/{fa
sta}"
      output:
      "SMETANA/{fasta}"
      shell:
      """
      set +u;source activate concoct_env;set -u
      mkdir -p SMETANA/$(basename {input})
      cd $TMPDIR
      cp
/c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe/carvemeOut-GFwM3
/$(basename {input})*
/c3se/NOBACKUP/groups/c3-c3se605-17-8/projects_francisco/.conda/envs/
concoct_env/lib/python2.7/site-packages/carveme/data/input/media_db.t
sv .
      smetana --detailed --flavor fbc2 --mediadb media_db.tsv -m
M1,M2,M3,M4,M5,M7,M8,M9,M10,M11,M13,M14,M15A,M15B,M16 $(basename
{input})* --solver CPLEX -o $(basename {input}) -v
      cp ERR*.tsv
/c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe/SMETANA/$(basena
me {input})
      """
```

The SMETANA results were summarized and parsed using the smetanaSummary rule below. For each interacting member, average values for SCS, MUS, and SMETANA were calculated for each media and for each metabolite class.

```
rule smetanaSummary:
      shell:
      """
      cd
"/c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe/SMETANA/"
```

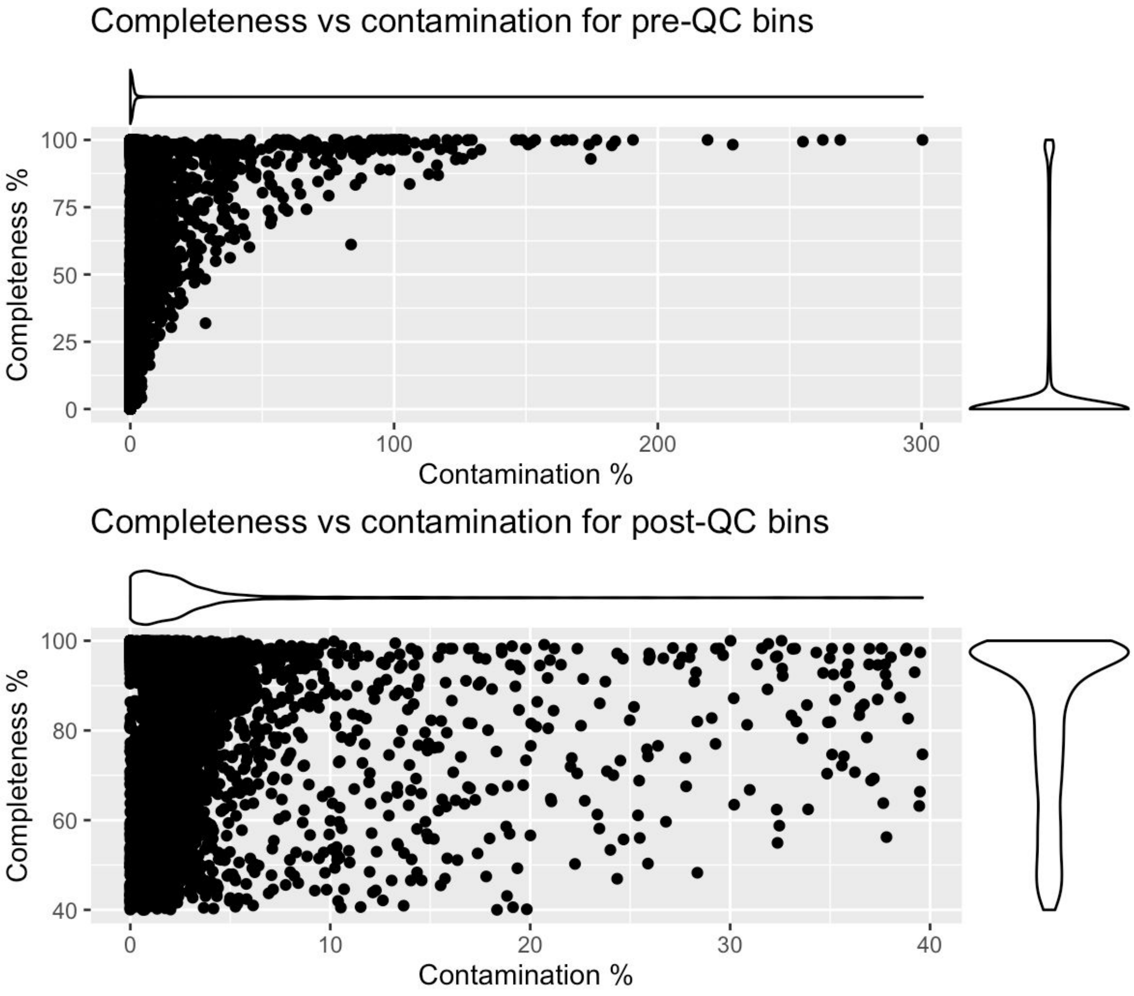
```
     awk 'NF'
"/c3se/NOBACKUP/users/zorrilla/binning/metabolismPipe/SMETANA/ERR2601
37/ERR260137_detailed.tsv"|head -n 1 >> SMETANA.sum
     for folder in ERR*;do
          awk 'NF' $folder/ERR*.tsv|tail -n +2 >> SMETANA.sum
     done
     cut -f1 --complement SMETANA.sum  >> SMETANA.summary
     rm SMETANA.sum
     for column in {2,3};do
          awk 'NF' SMETANA.summary|tail -n +2|cut -f$column >>
SMETANA.ids
     done
     less SMETANA.ids |sort -r|uniq >> SMETANA.binIds
     rm SMETANA.ids
     while read bin ;do
          echo -n "$bin" >> SMETANA.scores
          echo -n $'\t' >> SMETANA.scores
          N=$(less SMETANA.summary|grep "$bin"|wc -l)
          echo -n "$N" >> SMETANA.scores
          echo -n $'\t' >> SMETANA.scores
          SCS=$(less SMETANA.summary|grep "$bin"|awk '{sum+=$(NF-3)}
END {printf sum}');
          awk -v scs="$SCS" -v n="$N" 'BEGIN{print scs/n}' >>
SMETANA.scores
          echo -n $'\t' >> SMETANA.scores
          MUS=$(less SMETANA.summary|grep "$bin"|awk '{sum+=$(NF-2)}
END {printf sum}');
          awk -v mus="$MUS" -v n="$N" 'BEGIN{print mus/n}' >>
SMETANA.scores
          echo -n $'\t' >> SMETANA.scores
          MPS=$(less SMETANA.summary|grep "$bin"|awk '{sum+=$(NF-1)}
END {printf sum}');
          awk -v mps="$MPS" -v n="$N" 'BEGIN{print mps/n}' >>
SMETANA.scores
          echo -n $'\t' >> SMETANA.scores
          SMETANA=$(less SMETANA.summary|grep "$bin"|awk '{sum+=$NF}
END {printf sum}');
          awk -v smetana="$SMETANA" -v n="$N" 'BEGIN{print
smetana/n}' >> SMETANA.scores
          unset SCS MUS MPS SMETANA N
     done < SMETANA.binIds
     """
```

4.10 Linear models

Linear models were constructed using the summarized SMETANA results and abundance estimates described in previous sections. The linear models were formulated per species across samples, according to the equation below:

$$Abundance \sim \sum MUS + \sum SCS + \sum SMETANA + nInteractions \tag{Equation 2}$$

5 Supplementary Figures



Completeness vs contamination for pre-QC bins

Completeness vs contamination for post-QC bins

Supplementary Figure 1: Completeness vs contamination scatter with marginal violin plots for pre and post QC bins based on CheckM

Acknowledgements

I would like to give special thanks to my supervisor Aleksej Zelezniak, for giving me the opportunity to work under his supervision, guiding me in the process of learning and applying new ideas in creative ways, and for teaching me best practices in computational biology research. Thanks to Filip Buric for helping me with technical problems and questions, however big or small. Thanks to Johannes Alneberg, for his kind support and discussion regarding binning results and performance. Thanks to Daniel Machado, for helping with all things regarding genome-scale metabolic models and for providing me with useful supplementary files. Thanks to Angelo Limeta, Philip Gorter de Vries, Raphael Ferreira, and Andrea Clausen Lind for providing feedback and engaging in lively discussion.

References

[1] Escobar-Zepeda, Alejandra, et al. “The Road to Metagenomics: From Microbiology to DNA Sequencing Technologies and Bioinformatics.” *Frontiers in Genetics*, vol. 6, 2015, doi:10.3389/fgene.2015.00348.

[2] 23andMe. “DNA Genetic Testing & Analysis.” *23andMe*, https://www.23andme.com/?mdc2=true.

[3] “DNA Sequencing Costs: Data.” *Genome.gov*, https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data.

[4] Durack, Juliana, and Susan V. Lynch. “The Gut Microbiome: Relationships with Disease and Opportunities for Therapy.” *The Journal of Experimental Medicine*, vol. 216, no. 1, 2018, pp. 20–40., doi:10.1084/jem.20180448.

[5] Almeida, Alexandre, et al. “A New Genomic Blueprint of the Human Gut Microbiota.” *Nature*, vol. 568, no. 7753, Nov. 2019, pp. 499–504., doi:10.1038/s41586-019-0965-1.

[6] Nayfach, Stephen, et al. “New Insights from Uncultivated Genomes of the Global Human Gut Microbiome.” *Nature*, vol. 568, no. 7753, 2019, pp. 505–510., doi:10.1038/s41586-019-1058-x.

[7] Pasolli, Edoardo, et al. “Extensive Unexplored Human Microbiome Diversity Revealed by Over 150,000 Genomes from Metagenomes Spanning Age, Geography, and Lifestyle.” *Cell*, vol. 176, no. 3, 2019, doi:10.1016/j.cell.2019.01.001.

[8] Alneberg, Johannes, et al. “Binning Metagenomic Contigs by Coverage and Composition.” *Nature Methods*, vol. 11, no. 11, 2014, pp. 1144–1146., doi:10.1038/nmeth.3103.

[9] Thiele, Ines, and Bernhard Ø Palsson. “A Protocol for Generating a High-Quality Genome-Scale Metabolic Reconstruction.” *Nature Protocols*, vol. 5, no. 1, 2010, pp. 93–121., doi:10.1038/nprot.2009.203.

[10] Agren, Rasmus, et al. “The RAVEN Toolbox and Its Use for Generating a Genome-Scale Metabolic Model for Penicillium Chrysogenum.” *PLoS Computational Biology*, vol. 9, no. 3, 2013, doi:10.1371/journal.pcbi.1002980.

[11] Machado, Daniel, et al. “Fast Automated Reconstruction of Genome-Scale Metabolic Models for Microbial Species and Communities.” *Nucleic Acids Research*, vol. 46, no. 15, 2018, pp. 7542–7553., doi:10.1093/nar/gky537.

[12] Uritskiy, Gherman V., et al. “MetaWRAP—a Flexible Pipeline for Genome-Resolved Metagenomic Data Analysis.” *Microbiome*, vol. 6, no. 1, 2018, doi:10.1186/s40168-018-0541-1.

[13] Tramontano, Melanie, et al. “Nutritional Preferences of Human Gut Bacteria Reveal Their Metabolic Idiosyncrasies.” *Nature Microbiology*, vol. 3, no. 4, 2018, pp. 514–522., doi:10.1038/s41564-018-0123-9.

[14] Karlsson, Fredrik H., et al. “Gut Metagenome in European Women with Normal, Impaired and Diabetic Glucose Control.” *Nature*, vol. 498, no. 7452, 2013, pp. 99–103., doi:10.1038/nature12198.

[15] Zmora, Niv, et al. “Personalized Gut Mucosal Colonization Resistance to Empiric Probiotics Is Associated with Unique Host and Microbiome Features.” *Cell*, vol. 174, no. 6, 2018, doi:10.1016/j.cell.2018.08.041.

[16] Emiola, Akintunde, and Julia Oh. “High Throughput in Situ Metagenomic Measurement of Bacterial Replication at Ultra-Low Sequencing Coverage.” *Nature Communications*, vol. 9, no. 1, 2018, doi:10.1038/s41467-018-07240-8.

[17] Chen, Shifu, et al. “Fastp: an Ultra-Fast All-in-One FASTQ Preprocessor.” *Bioinformatics*, vol. 34, no. 17, Jan. 2018, pp. i884–i890., doi:10.1093/bioinformatics/bty560.

[18] Li, Dinghua, et al. “MEGAHIT: an Ultra-Fast Single-Node Solution for Large and Complex Metagenomics Assembly via Succinct De Bruijn Graph.” *Bioinformatics*, vol. 31, no. 10, 2015, pp. 1674–1676., doi:10.1093/bioinformatics/btv033.

[19] Bray, Nicolas L, et al. “Near-Optimal Probabilistic RNA-Seq Quantification.” *Nature Biotechnology*, vol. 34, no. 5, Apr. 2016, pp. 525–527., doi:10.1038/nbt.3519.

[20] Parks, Donovan H, et al. “CheckM: Assessing the Quality of Microbial Genomes Recovered from Isolates, Single Cells, and Metagenomes.” 2015, doi:10.7287/peerj.preprints.554.

[21] Li, H., and R. Durbin. “Fast and Accurate Short Read Alignment with Burrows-Wheeler Transform.” *Bioinformatics*, vol. 25, no. 14, 2009, pp. 1754–1760., doi:10.1093/bioinformatics/btp324.

[22] Milanese, Alessio, et al. “Microbial Abundance, Activity and Population Genomic Profiling with mOTUs2.” *Nature Communications*, vol. 10, no. 1, Apr. 2019, doi:10.1038/s41467-019-08844-4.

[23] Lieven, Christian, et al. “Memote: A Community Driven Effort Towards a Standardized Genome-Scale Metabolic Model Test Suite.” *BioRxiv*, June 2018, doi:https://doi.org/10.1101/350991 .

[24] Zelezniak, Aleksej, et al. "Metabolic dependencies drive species co-occurrence in diverse microbial communities." Proceedings of the National Academy of Sciences 112 (2015): 6449-454.