In silico modelling of the effect of bacterial DNA fragmentation on gene counts using digital droplet PCR

Bachelor Thesis

Soil & Water Research Infrastructure and Institute of Soil Biology

ANAEROBIC AND MOLECULAR MICROBIOLOGY LAB

Sebastian Vranjes

Supervisor: Angel Roey, Dr. rer. Nat.

Bibliography:

Vranjes, S, 2022 *In silico* modelling of the effect of bacterial DNA fragmentation on gene counts using digital droplet PCR. Bc., Thesis, in English -21p., Faculty of Science, University of South Bohemia, Ceske Budejovice, Czech Republic.

Annotation:

The aim of this study is to establish a correction factor for ddPCR, accounting for instances of the target sequence's occurrences being miscounted due to endpoint detection limitations.

Affirmation:

I declare that I am the author of this qualification thesis and that in writing it I have used the sources and literature displayed in the list of used sources only.

Ceske Budejovice 16.08.2022 Signature: Sebastian Vranjes

Abstract

Accurate measurement of microbial population sizes in the environment is key towards developing quantitative and predictive models in microbial ecology.

Quantitative PCR (qPCR) is a common technique for quantifying genes and transcripts, but is limited in its use due to reliance on an external standard curve as well as its susceptibility to inhibitors. Digital PCR (dPCR), on the other hand, is a more recently developed method that is unaffected by these issues. However, it possesses a lower dynamic range, higher cost per sample, and potential inaccuracies due to its use of endpoint analysis for the interpretation of results.

This work aims to model the miscounting of hits in digital droplet PCR (ddPCR), one of the most common variants of dPCR, due to endpoint detection limitations, and to establish a correction factor to account for it. This is done in addition to the common Poisson distribution correction, which accounts only for the presence of multiple DNA templates in a droplet.

To achieve this, the following *in silico* model was constructed: genomes containing a specific gene of interest (16S RNA) were randomly obtained from a public database, screened for quality, and then fragmented in a way that mimics real DNA extraction using a skewed normal distribution. The obtained fragments were then amplified via *in silico* PCR and the number of hits compared to the number of genes on the whole genome to quantify the mismatches.

The results show a significant undercounting of the real number of templates by nearly 20%, depending on the amplicon length. Fortunately, the bias appears consistent over multiple runs, hence a correction factor for future ddPCR runs could be established.

Table of Contents:

1. Introduction

1.1 Bacterial cells

Prokaryotes (bacteria and archaea) are unicellular organisms that are present in nearly all of Earth's habitats. They are morphologically varied, with forms ranging from rods and spirals to spheres. Their length spans from 0.1 micrometers to 0.7 millimeters among the larger species [1]. In comparison to eukaryotic cells, prokaryotes lack a nucleus, mitochondria, and chloroplast. Their DNA is present within the cytoplasm as a single circular bacterial chromosome referred to as the nucleoid [2-4]. The DNA of prokaryotes is typically present as a single circular molecule (with the exception of plasmids, which function as additional circular DNA molecules within the cell and contain only non-essential genes). Due to this shape, the prokaryotic DNA replicates in a semi-conservative manner [5,6]. Additionally, prokaryotic cells have the highest reproduction rate of any organism under beneficial conditions, with a doubling time of 17 minutes among the quickest species [7].

Thanks to the ease of manipulation, as well as the insight it offers into pathogens, prokaryotic genomes represent one of the most popular targets of study, allowing for an abundance of their DNA to be used in research [8].

1.2 The small subunit of the ribosomal RNA as a marker gene (16S rRNA gene)

The small subunit of the ribosomal RNA (16S rRNA in prokaryotes) is a component of the 30S subunit of a prokaryotic ribosome. Functionally, it provides most of the 30S structure and binds to the Shine-Dalgarno sequence (SDS), which is a portion of mRNA located approximately 8 base pairs upstream from the start codon [9].

The 30S subunit itself is a part of the 70S ribosome found in prokaryotes and acts as an integral component for mRNA translation by binding to 3 prokaryotic initiation factors. The 16S rRNA directs the initiating start codon of the mRNA to the correct position using the SDS, thus ensuring that translation will begin from the correct location. The strength of the bond between the SDS on the mRNA and the 16S rRNA determines the efficiency of the translation that will take place [10-14].

The 16S rRNA gene is commonly used as a phylogenetic marker for classifying bacteria and archaea. This is thanks to two attributes of the gene: 1. the gene is present in all living organisms. 2. due to its central function in protein synthesis it is highly conserved and diverged relatively little throughout evolution [15].

The small-subunit rRNA (16S and 18S) was used by Carl Woese to distinguish archaea from the remaining two domains by sequencing its nucleotide composition and comparing it among various eukaryotes and prokaryotes. This led to the current three-domain model of the tree of life (AKA the Woesian tree of life) [16].

Thanks to its universality the 16S rRNA gene is also used to identify and quantify microorganisms in the environment, However, because many species of bacteria possess variable copy numbers of the 16S rRNA gene a simple conversion of gene copies to cell numbers cannot be made. Hence, a correction factor is needed when trying to deduce genome or cell number from the number of 16S rRNA gene copies [17].

1.3 Quantitative PCR (qPCR)

PCR (polymerase chain reaction) is a molecular biology technique that is used to make millions to billions of copies from scarce amounts of target DNA [18]. It was invented in 1983 by American biochemist Kary Mullis and has since become one of the main molecular laboratory research techniques for numerous applications including criminal forensics and biomedical research [19]. It functions by using two components, first a DNA oligonucleotide (primer) complimentary to the beginning of the target sequence, and then a thermostable DNA polymerase for the replication [20,21]. Using these two components, the system starts the temperature cycle at 95-100 °C in order to split the double stranded DNA into singular strands, which is also known as the denaturation or melting step, and then the temperature drops to 45-55 °C to allow the primers to anneal the singled stranded DNA (ssDNA). In the final step of the cycle, the new strand is synthesized via the DNA polymerase enzyme at approximately 75 °C [22].

The laboratory technique called qPCR, also known as real-time PCR, utilizes the principle of PCR but also monitors the amplification of the desired DNA molecule in real time. This method is useful for deducing the bacterial load in a sample since bacteria only contain one DNA and

the DNA count can be directly correlated to the cell quantity [23]. It does so by either using a sequence-specific DNA probe made up of oligonucleotides and labelled with a fluorescent reporter, or a non-specific fluorescent dye that will intercalate with any double-stranded DNA (typically SYBR Green) [24,25]. One drawback of the method utilizing the fluorescent dye can result in the dye binding to unwanted components of the PCR process such as primer dimers or genomic DNA, while the fluorescent reporter will only be detected following the hybridization of the DNA probe with its complimentary sequence [26]. Real-time PCR has a wide variety of applications in the laboratory, most commonly within diagnostic and basic research [27,28]. Industrially, the technique is typically used for the quantification of microbial load in foods, the detection of GMO's (Genetically modified organisms), and in quantification and genotyping of human pathogens [29-31].

1.4 Digital PCR (dPCR)

Digital PCR (dPCR) is a type of digital PCR where a DNA sample is distributed into numerous individual partitions (from hundreds to even millions in advanced systems) [32]. The sample is either separated in thousands of water-in-oil droplets in one tube using a combination of microfluidics and proprietary surfactant chemistries (in which case the method is called digital droplet PCR, or ddPCR), or distributed into nanowells on a chip. In both cases, the result is several thousand PCR reactions occurring in parallel for one sample. The data is then analysed by the system and the fraction of PCR positive droplets in the original sample is calculated [33].

Since the number of templates in each droplet is random, the Poisson distribution formula can be applied to make a prediction of how many of the positive droplets contained multiple templates, and thus attempt to predict the target DNA template concentration within the original sample. The equation for the Poisson distribution formula can be found below:

$$\lambda = -ln(1 - p)$$

(Where "p" represents the number of positive droplets and "λ" refers to the amount of target copies per droplet) [34].

Compared to qPCR, ddPCR boasts many advantages, such as no reliance on an external standard and reference samples, improved performance for applications requiring higher sensitivity and precision, as well as a higher resistance to inhibitors, and a lower coefficient of variation. However, ddPCR also possesses a lower sample throughput than qPCR, as well as a higher cost per sample, and a smaller dynamic range [35,36]. Currently, the preparation of a sample for ddPCR involves the extraction of DNA from a cell. This process typically results in the DNA being fragmented in lengths of 2K – 20K Base pairs that may result in one of the two following situations. Firstly, since as previously mentioned many bacterial species may contain multiple copies of the 16S rRNA gene (up to 15 !), more than one copy may end up on one fragment but be counted only once. This is because ddPCR uses an endpoint detection method to quantify results (either a positive or negative response) and does not account for the signal strength (which is positively correlated with the number of copies in the droplet, but also negatively correlated with the presence of PCR inhibitors) [34]. The second more likely situation is that the DNA might be fragmented in the region of a 16S rRNA gene resulting in an underestimation of the total gene count due to a reduced number of viable targets. This shows the need for a correction factor to be calculated for ddPCR, accounting for instances where multiple complete target sequences, or incomplete instances of the target sequence are falsely registered as an individual or incomplete hit rather than their number of occurrence.

2. Aim

The aim of this study is to establish a correction factor for ddPCR, accounting for instances of the target sequence's occurrences being miscounted due to endpoint detection limitations.

2.1 Objectives

- Data sets are extracted from the number of genes of interest present in the downloaded genomes
- Data sets are compiled from the *in silico* fragmented genomes showing whether the 16S RNA gene was maintained.
- Data from both sets will be compared to establish a correction factor for differing values.

3. Code

3.1 Code Process

Below is the flowchart that describes the process executed by the written code (Fig. 1).

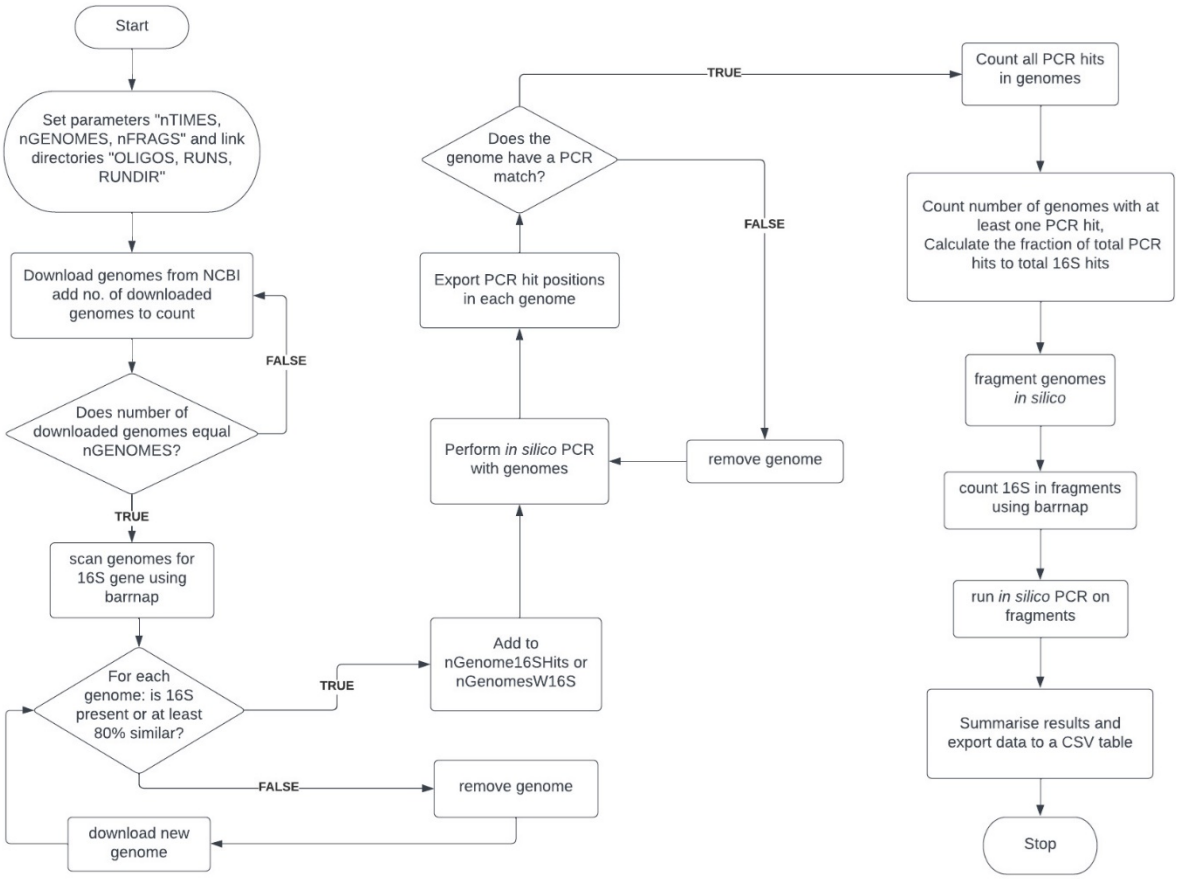


Figure 1. Code diagram for modelling ddPCR underestimation.

A full version of the source code for the wrapper, calculation and plotting scripts are available in the Appendix, as well as online on GitHub.

The algorithm performs as follows:

1. First, basic parameters such as nTIMES (number of runs), nGENOMES (number of genomes), and nFRAGS (number of fragments) are set and the paths to the necessary directories are created.

2. Next, prokaryotic genomes are downloaded from theGenBank database (NCBI) randomly until the number of downloaded genomes equals the specified parameter in the first step. The program also

keeps track of the index numbers of each downloaded genome to avoid selecting the same genome twice.

3. The genomes are then scanned for the presence of the 16S gene using the tool barrnap. Once the scan is completed, the genomes that have less than 80% of the 16S gene present are removed, and new ones downloaded in their place and the process repeated until no genomes need to be removed.

4. Once the previous process has been completed, the number of genomes containing the 16S gene (nGenomesW16S) and the number of total hits within those genomes (nGenome16Shits) are saved.

5. The script then runs *in silico* PCR on the genomes using primers designed to amplify the target gene, saving the results in a separate folder. The genomes are then read and cross-referenced with the hit positions from the PCR results to check for mismatches. If a genome has no PCR match it is removed from the genome pool and the *in silico PCR* is performed again until all genomes have a PCR match.

6. The total number of PCR hits, as well as the number of genomes containing at least one PCR hit are counted, and the fraction of total PCR hits to total 16S (nGenome16Shits) is calculated.

7. Once the values are calculated, the genomes are fragmented *in silico* using SIPSim and the number of 16S genes in the fragments counted. The fragments then also undergo *in silico* PCR and the number of hits in both cases are saved. This last step mimics ddPCR.

8. The results are then summarized, and the data exported to a CSV table for analysis.

4. Results

The program was run 100 times, each with 1000 genomes (mimicking a complex environmental community) and each genome was fragmented to 1000 pieces. The model therefore screened a total of 100 × 10^6 fragments. Figure 2 illustrates the resulting fragmentation from 2 out of the 100 runs to show the variability in taxonomy and G+C content of the selected bacteria.

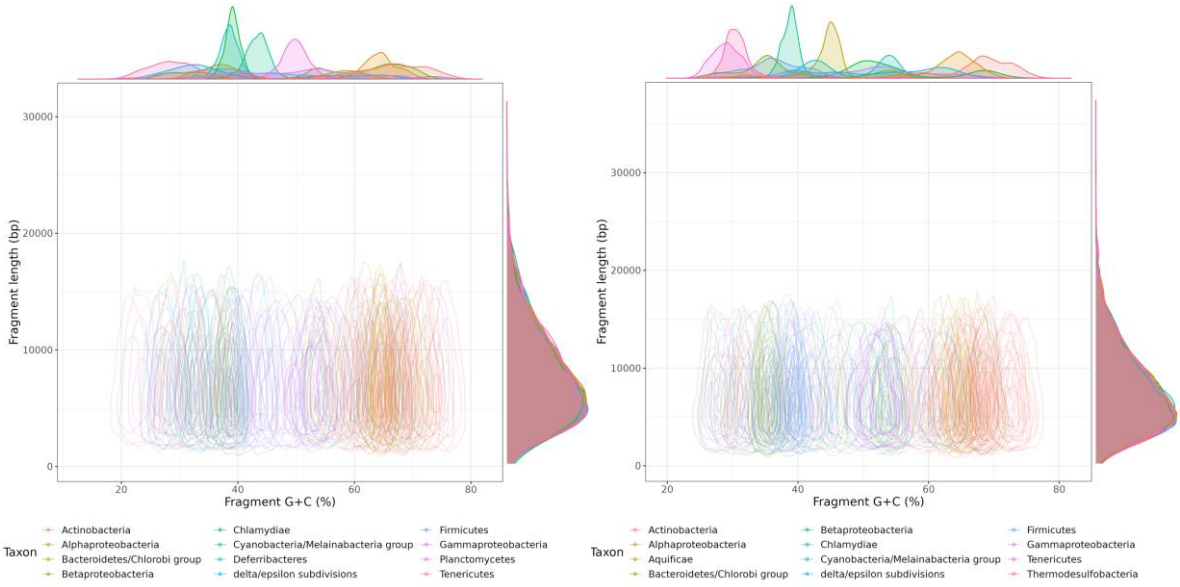


Figure 2. An example of the distribution of genome taxonomy, DNA fragment lengths and G+C contents of 1000 genomes obtained in two different model runs.

The following data was obtained with Table 1 containing the values obtained prior to the digital PCR, and Table 2 contains the results after the digital PCR. A select number of runs from the results was extracted and compiled in the tables below to demonstrate the format of the programs output.

Table 1. Results of 10 different runs

Mean 16S rRNA genes per genome	Frags w. 16S	P frags w. 16S	Frags w. PCR (ddPCR outcome)	P frags w. PCR	Frags w. multi 16S	P frags w. multi 16S
4.4	10443	0.0104	8950	0.857	287	0.0275
4.4	10596	0.0106	9020	0.8513	242	0.0228
4.4	10784	0.0108	9244	0.8572	238	0.0221
4.4	10791	0.0108	9256	0.8578	201	0.0186
4.5	10503	0.0105	9030	0.8598	227	0.0216
4.4	10396	0.0104	8947	0.8606	223	0.0215
4.3	10437	0.0104	8923	0.8549	207	0.0198
4.6	10760	0.0108	9226	0.8574	269	0.025
4.3	10437	0.0104	8923	0.8549	207	0.0198
4.5	10428	0.0104	8936	0.8569	192	0.0184

Table 2:

Frags w. multi PCR	P frags w. multi PCR	ddPCR correction factor	Corrected ddPCR value
213	0.0238	1.191	10663
159	0.0176	1.193	10758
140	0.0151	1.182	10926
146	0.0158	1.182	10943
168	0.0186	1.182	10672
142	0.0159	1.178	10541

Frags w PCR (ddPCR outcome) is the number of hits expected to be returned from a standard ddPCR procedure. Frags w. multi 16S represents the number of fragments that contain more than one 16S gene (or gene fragment) while Frags w. multi PCR refers to fragments with multiple 16S genes where the 16S fragments are long enough to be amplified using PCR with the chosen primers.
The P prefixing some columns refers to the proportion, or fraction of the genomes or fragments, to their total number.

For the ddPCR correction factor the equation x was used:

$$Correction\ factor = \frac{1}{Genes\ split} + \frac{Redundant\ hits}{Total\ PCR\ hits}$$

The correction factor was then used to provide the data in the column labeled Corrected ddPCR value.

The averages of the obtained values from 100 runs were compiled and are listed below in Table 3:

Table 3:

	Average	Standard deviation
16S per Genome	4.407	0.085582
Frags w. 16S	10508.11	225.7658
P. frags w. 16S	0.010502	0.000229
Frags w. PCR (ddPCR outcome)	8996.15	199.0591
P. frags w. PCR	0.85611	0.003635
Frags w. multi 16S	230.71	25.75302
P. frags w. multi 16S	0.02195	0.002372
Frags w. multi PCR	158.04	20.81808
P. frags w. multi PCR	0.017558	0.002236
ddPCR correction factor	1.18616	0.005281
Corrected ddPCR value	10670.79	232.5848

5. Discussion

The use of ddPCR in molecular biology labs is rapidly increasing with optimization to accuracy and the user-friendly improvements being developed [37].

As a relatively new technology brought to the market in 2012 not all biases and caveats are known, and with its rising popularity it is imperative to optimize the process by eliminating potential inaccuracies in data, as these could affect a large amount of research that makes use of this tool. This holds especially true for the field of environmental microbiology, which is moving towards more quantitative research that would therefore require the highest possible accuracy.

Here we presented and modelled a persistent bias caused by the end-point detection system, namely the occurrence of partial, or multiple template molecules within one droplet being miscounted as 0, or 1 ddPCR hit. As previously stated, the aim of this study was to establish a correction factor for ddPCR, which would account for this persistent bias. An example of this would be with malaria parasites where late trophozoites and schizonts contain several genomes or in general cases where DNA is fragmented during sample preparation so that the genome of interest is split [36]. The

current solution to this issue is the application of a Poisson model to calculate the number of times multiple fragments are present within a single droplet but doesn't estimate the overall occurrence of this miscount or partial template molecules not being counted [38]. The data that was obtained in this study yielded an average ddPCR correction factor of 1.18616, showing an approximately 18.6% deficit in the number of hits. Since the deficit exceeds 10% the correction factor can be considered significant regarding data interpretation, impacting the obtained results. With this data the accuracy, robustness, and reproducibility of ddPCR can hopefully be increased to ensure that it sees more common application in clinical trials and testing [38]. This could impact research in areas where ddPCR is used for quantitative analysis, such as in the identification of mixed meat products and quantification of their constituents, or the accurate quantification of parasites where the need for an external standard curve would make it difficult to compare research [36, 39].

5.1 Limitations

Although the results show a sizeable deficit in the number of hits in ddPCR results, the study only performs the fragmentation and PCR *in silico*, which could potentially mean that actual fragmentation and ddPCR would be impacted by the miscounting of hits to a different degree from what was determined. However, the size of the deficit does show the need for such a correction factor to be established, which will hopefully be supported by *in vitro* research contributions in the future.

The effect also depends on: DNA extraction method (i.e. DNA fragmentation), gene of interest and the primers used (or more correctly the length of the amplicon).

6. Conclusion

In conclusion, the task was to establish a correction factor for ddPCR caused by endpoint detection limitations. This was achieved via *in silico* fragmentation of genomes confirmed to contain the 16S RNA gene before being fractured, and then comparing the number of ddPCR hits from the fragments with the number of genes of interest in the whole genomes. Using the data obtained, a correction factor could be established which would account for the number of miscounted genes of interest among the fragments. The obtained correction factor had a value of 1.18616 which shows a hit deficit of over 18%, therefore having a significant impact on obtained data. This correction factor can assist in bolstering the reliability of quantitative ddPCR research or applications in the future, and hopefully allow ddPCR as an analytical method to see more application in the field with increased reproducibility.

7. Literature

1. Madigan, C. A., Cambier, C., Kelly-Scumpia, K. M., Scumpia, P. O., Cheng, T. Y., Zailaa, J., Bloom, B. R., Moody, D. B., Smale, S. T., Sagasti, A., Modlin, R. L., & Ramakrishnan, L. (2017). A Macrophage Response to Mycobacterium leprae Phenolic Glycolipid Initiates Nerve Damage in Leprosy. *Cell*, *170*(5), 973–985.e10. https://doi.org/10.1016/j.cell.2017.07.030

2. Flemming, Hans-Curt, and Stefan Wuertz. “Bacteria and Archaea on Earth and Their Abundance in Biofilms.” *Nature Reviews Microbiology*, vol. 17, no. 4, 2019, pp. 247–60. *Crossref*, doi:10.1038/s41579-019-0158-9.

3. Cooper, R., and J. C. Lawrence. “Micro-Organisms and Wounds.” *Journal of Wound Care*, vol. 5, no. 5, 1996, pp. 233–36. *Crossref*, doi:10.12968/jowc.1996.5.5.233.

4. Williams C (2011). "Who are you calling simple?". New Scientist. **211** (2821): 38–41. doi:10.1016/S0262-4079(11)61709-0.

5. Thanbichler, Martin, et al. “The Bacterial Nucleoid: A Highly Organized and Dynamic Structure.” *Journal of Cellular Biochemistry*, vol. 96, no. 3, 2005, pp. 506–21. *Crossref*, doi:10.1002/jcb.20519.

6. Streips, Uldis, and Ronald Yasbin. *Modern Microbial Genetics*. 2nd ed., Wiley-Liss, 2002.

7. *Fundamentals Of Microbiology 10Th Edition*. 10th ed., Jones & Bartlett Learning, 2021.

8. Donkor, Eric. “Sequencing of Bacterial Genomes: Principles and Insights into Pathogenesis and Development of Antibiotics.” *Genes*, vol. 4, no. 4, 2013, pp. 556–72. *Crossref*, doi:10.3390/genes4040556.

9. Steitz, J. A., and K. Jakes. “How Ribosomes Select Initiator Regions in MRNA: Base Pair Formation between the 3’ Terminus of 16S RRNA and the MRNA during Initiation of Protein Synthesis in Escherichia Coli.” *Proceedings of the National Academy of Sciences*, vol. 72, no. 12, 1975, pp. 4734–38. *Crossref*, doi:10.1073/pnas.72.12.4734.

10. Schluenzen, Frank, et al. “Structure of Functionally Activated Small Ribosomal Subunit at 3.3 Å Resolution.” *Cell*, vol. 102, no. 5, 2000, pp. 615–23. *Crossref*, doi:10.1016/s0092-8674(00)00084-2.

11. Gold, L., et al. “Translational Initiation in Prokaryotes.” *Annual Review of Microbiology*, vol. 35, no. 1, 1981, pp. 365–403. *Crossref*, doi:10.1146/annurev.mi.35.100181.002053.

12. Malys, Naglis. “Shine-Dalgarno Sequence of Bacteriophage T4: GAGG Prevails in Early Genes.” *Molecular Biology Reports*, vol. 39, no. 1, 2011, pp. 33–39. *Crossref*, doi:10.1007/s11033-011-0707-4.

13. Gualerzi, Claudio O., and Cynthia L. Pon. “Initiation of MRNA Translation in Prokaryotes.” *Biochemistry*, vol. 29, no. 25, 1990, pp. 5881–89. *Crossref*, doi:10.1021/bi00477a001.

14. Igarashi, Kazuei, et al. “On the Aminoacyl-TRNA Binding Site of the 30-S Ribosomal Subunit and Its Relation to the Chain Initiation Site of the Ribosome.” *Biochimica et Biophysica Acta (BBA) - Nucleic Acids and Protein Synthesis*, vol. 228, no. 3, 1971, pp. 728–31. *Crossref*, doi:10.1016/0005-2787(71)90737-4.

15. Weisburg, W. G., et al. “16S Ribosomal DNA Amplification for Phylogenetic Study.” *Journal of Bacteriology*, vol. 173, no. 2, 1991, pp. 697–703. *Crossref*, doi:10.1128/jb.173.2.697-703.1991.

16. Woese, C. R., et al. “Towards a Natural System of Organisms: Proposal for the Domains Archaea, Bacteria, and Eucarya.” *Proceedings of the National Academy of Sciences*, vol. 87, no. 12, 1990, pp. 4576–79. *Crossref*, doi:10.1073/pnas.87.12.4576.

17. Case, Rebecca J., et al. “Use of 16S RRNA and RpoB Genes as Molecular Markers for Microbial Ecology Studies.” *Applied and Environmental Microbiology*, vol. 73, no. 1, 2007, pp. 278–88. *Crossref*, doi:10.1128/aem.01177-06.

18. Saiki, RK, et al. “Primer-Directed Enzymatic Amplification of DNA with a Thermostable DNA Polymerase.” *Science*, vol. 239, no. 4839, 1988, pp. 487–91. *Crossref*, doi:10.1126/science.239.4839.487.

19. Shampo, Marc A., and Robert A. Kyle. “Kary B. Mullis—Nobel Laureate for Procedure to Replicate DNA.” *Mayo Clinic Proceedings*, vol. 77, no. 7, 2002, p. 606. *Crossref*, doi:10.4065/77.7.606.

20. Lawyer, F. C., et al. “High-Level Expression, Purification, and Enzymatic Characterization of Full-Length Thermus Aquaticus DNA Polymerase and a Truncated Form Deficient in 5’ to 3’ Exonuclease Activity.” *Genome Research*, vol. 2, no. 4, 1993, pp. 275–87. *Crossref*, doi:10.1101/gr.2.4.275.

21. Cox, Michael, et al. *Molecular Biology: Principles and Practice*. Second, W. H. Freeman, 2015.

22. Porta, Angela R., and Edward Enners. “Determining Annealing Temperatures for Polymerase Chain Reaction.” *The American Biology Teacher*, vol. 74, no. 4, 2012, pp. 256–60. *Crossref*, doi:10.1525/abt.2012.74.4.9.

23. Heid, C. A., et al. “Real Time Quantitative PCR.” *Genome Research*, vol. 6, no. 10, 1996, pp. 986–94. *Crossref*, doi:10.1101/gr.6.10.986.

24. Schmittgen, Thomas D. “Real-Time Quantitative PCR.” *Methods*, vol. 25, no. 4, 2001, pp. 383–85. *Crossref*, doi:10.1006/meth.2001.1260.

25. Logan, Julie, et al. *Real-Time PCR: Current Technology and Applications*. 1st ed., Caister Academic Press, 2009.

26. Lind, Kristina, et al. “Combining Sequence-Specific Probes and DNA Binding Dyes in Real-Time PCR for Specific Nucleic Acid Quantification and Melting Curve Analysis.” *BioTechniques*, vol. 40, no. 3, 2006, pp. 315–19. *Crossref*, doi:10.2144/000112101.

27. Espy, M. J., et al. “Real-Time PCR in Clinical Microbiology: Applications for Routine Laboratory Testing.” *Clinical Microbiology Reviews*, vol. 19, no. 1, 2006, pp. 165–256. *Crossref*, doi:10.1128/cmr.19.1.165-256.2006.

28. Dhamad, Ahmed E., and Muna A. Abdal Rhida. “COVID-19: Molecular and Serological Detection Methods.” *PeerJ*, vol. 8, 2020, p. e10180. *Crossref*, doi:10.7717/peerj.10180.

29. Filion, and Martin Filion. *Quantitative Real-Time PCR in Applied Microbiology*. Caister Academic Press Limited, 2012.

30. Holst-Jensen, Arne, et al. “PCR Technology for Screening and Quantification of Genetically Modified Organisms (GMOs).” *Analytical and Bioanalytical Chemistry*, vol. 375, no. 8, 2003, pp. 985–93. *Crossref*, doi:10.1007/s00216-003-1767-7.

31. Yeh, Shiou-Hwei, et al. “Quantification and Genotyping of Hepatitis B Virus in a Single Reaction by Real-Time PCR and Melting Curve Analysis.” *Journal of Hepatology*, vol. 41, no. 4, 2004, pp. 659–66. *Crossref*, doi:10.1016/j.jhep.2004.06.031.

32. Baker, Monya. “Digital PCR Hits Its Stride.” *Nature Methods*, vol. 9, no. 6, 2012, pp. 541–44. *Crossref*, doi:10.1038/nmeth.2027.

33. Pohl, Gudrun, and Ie-Ming Shih. “Principle and Applications of Digital PCR.” *Expert Review of Molecular Diagnostics*, vol. 4, no. 1, 2004, pp. 41–47. *Crossref*, doi:10.1586/14737159.4.1.41.

34. McDermott, Geoffrey P., et al. “Multiplexed Target Detection Using DNA-Binding Dye Chemistry in Droplet Digital PCR.” *Analytical Chemistry*, vol. 85, no. 23, 2013, pp. 11619–27. *Crossref*, doi:10.1021/ac403061n.

35. Pekin, Deniz, et al. “Quantitative and Sensitive Detection of Rare Mutations Using Droplet-Based Microfluidics.” *Lab on a Chip*, vol. 11, no. 13, 2011, p. 2156. *Crossref*, doi:10.1039/c1lc20128j.

36. Yang, Rongchang, et al. “Comparison of Next-Generation Droplet Digital PCR (DdPCR) with Quantitative PCR (QPCR) for Enumeration of Cryptosporidium Oocysts in Faecal Samples.” *International Journal for Parasitology*, vol. 44, no. 14, 2014, pp. 1105–13. *Crossref*, doi:10.1016/j.ijpara.2014.08.004.

37. *A quality control revolution for droplet digital PCR*. (2020). Nature. Retrieved July 16, 2022, from https://www.nature.com/articles/d42473-020-00123-x?error=cookies_not_supported&code=63a4663a-fc2c-4ab7-9ff2-139b8a5272a1

38. Wong, Y. K., Tsang, H. F., Xue, V. W., Chan, C. M., Au, T. C., Cho, W. C., Chan, L. W., & Wong, S. C. (2017). Applications of digital PCR in precision medicine. *Expert Review of Precision Medicine and Drug Development*, *2*(3), 177–186. https://doi.org/10.1080/23808993.2017.1347482

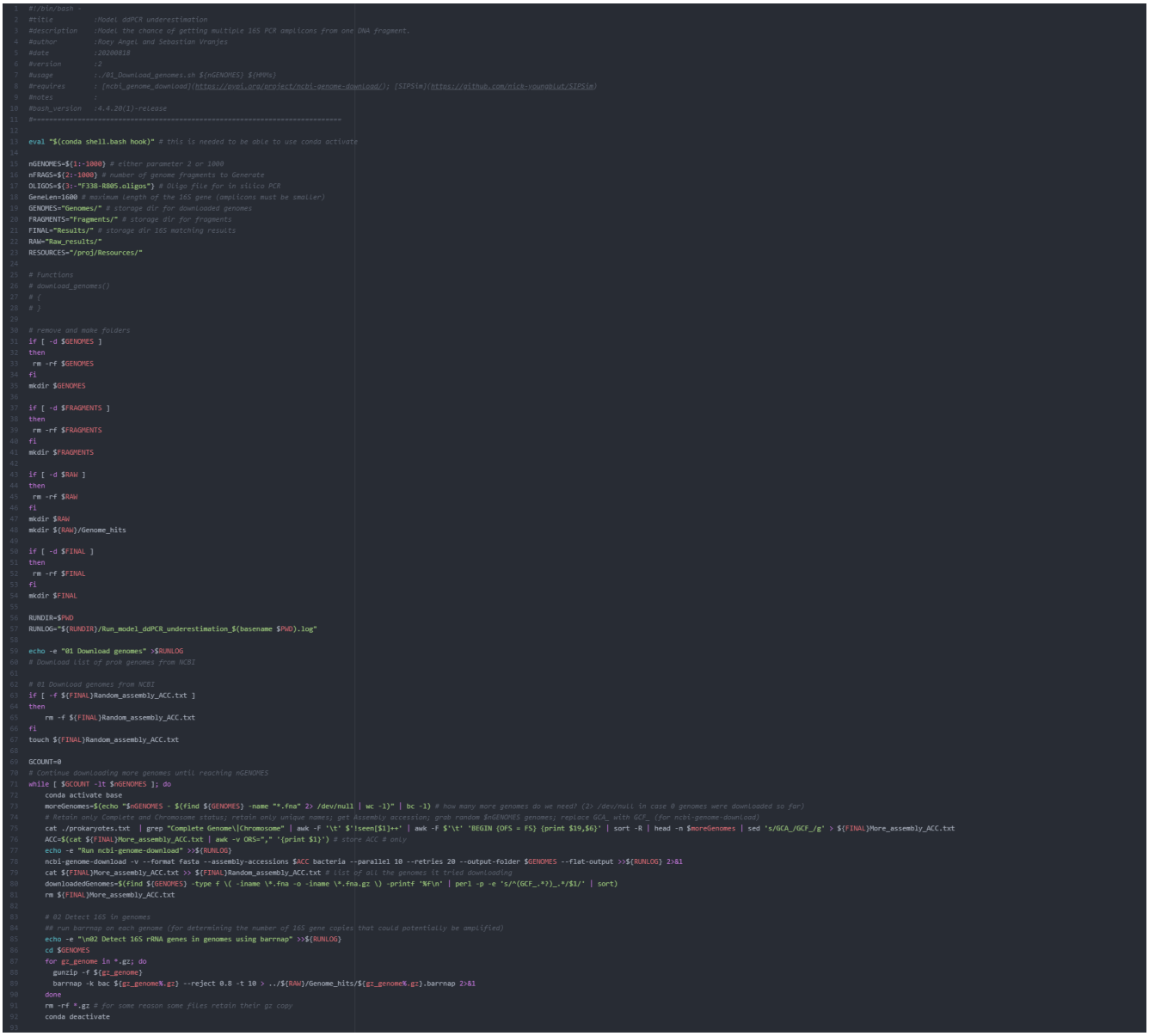
39. *Species identification and quantification in meat and meat products using droplet digital PCR (ddPCR)*. (2015, April 15). ScienceDirect. https://linkinghub.elsevier.com/retrieve/pii/S0308814614017129

8. Appendices

Link to GitHub: https://github.com/roey-angel/ddPCR-simulation

9. Shell script

These are the screenshots of the used scripts.



```
#!/bin/bash -
#title          :Model ddPCR underestimation
#description    :Model the chance of getting multiple 16S PCR amplicons from one DNA fragment.
#author         :Roey Angel and Sebastian Vranjes
#date           :20200818
#version        :2
#usage          :./01_Download_genomes.sh ${nGENOMES} ${MPMs}
#requires       : [ncbi_genome_download](https://pypi.org/project/ncbi-genome-download/); [SIPSim](https://github.com/nick-youngblut/SIPSim)
#notes          :
#bash_version   :4.4.20(1)-release
#============================================================================

eval "$(conda shell.bash hook)" # this is needed to be able to use conda activate

nGENOMES=${1:-1000} # either parameter 2 or 1000
nFRAGS=${2:-1000} # number of genome fragments to Generate
OLIGOS=${3:-"F338-R805.oligos"} # Oligo file for in silico PCR
GeneLen=1600 # maximum length of the 16S gene (amplicons must be smaller)
GENOMES="Genomes/" # storage dir for downloaded genomes
FRAGMENTS="Fragments/" # storage dir for fragments
FINAL="Results/" # storage dir 16S matching results
RAW="Raw_results/"
RESOURCES="/proj/Resources/"

# Functions
# download_genomes()
# {
# }

# remove and make folders
if [ -d $GENOMES ]
then
 rm -rf $GENOMES
fi
mkdir $GENOMES

if [ -d $FRAGMENTS ]
then
 rm -rf $FRAGMENTS
fi
mkdir $FRAGMENTS

if [ -d $RAW ]
then
 rm -rf $RAW
fi
mkdir $RAW
mkdir ${RAW}/Genome_hits

if [ -d $FINAL ]
then
 rm -rf $FINAL
fi
mkdir $FINAL

RUNDIR=$PWD
RUNLOG="${RUNDIR}/Run_model_ddPCR_underestimation_$(basename $PWD).log"

echo -e "01 Download genomes" >$RUNLOG
# Download list of prok genomes from NCBI

# 01 Download genomes from NCBI
if [ -f ${FINAL}Random_assembly_ACC.txt ]
then
    rm -f ${FINAL}Random_assembly_ACC.txt
fi
touch ${FINAL}Random_assembly_ACC.txt

GCOUNT=0
# Continue downloading more genomes until reaching nGENOMES
while [ $GCOUNT -lt $nGENOMES ]; do
    conda activate base
    moreGenomes=$(echo "$nGENOMES - $(find ${GENOMES} -name "*.fna" 2> /dev/null | wc -l)" | bc -l) # how many more genomes do we need? (2> /dev/null in case 0 genomes were downloaded so far)
    # Retain only Complete and Chromosome status; retain only unique names; get Assembly accession; grab random $nGENOMES genomes; replace GCA_ with GCF_ (for ncbi-genome-download)
    cat ./prokaryotes.txt  | grep "Complete Genome\|Chromosome" | awk -F '\t' '!seen[$11]++' | awk -F $'\t' 'BEGIN {OFS = FS} {print $19,$6}' | sort -R | head -n $moreGenomes | sed 's/GCA_/GCF_/g' > ${FINAL}More_assembly_ACC.txt
    ACC=$(cat ${FINAL}More_assembly_ACC.txt | awk -v ORS="," '{print $1}') # store ACC # only
    echo -e "Run ncbi-genome-download" >>${RUNLOG}
    ncbi-genome-download -v --format fasta --assembly-accessions $ACC bacteria --parallel 10 --retries 20 --output-folder $GENOMES --flat-output >>${RUNLOG} 2>&1
    cat ${FINAL}More_assembly_ACC.txt >> ${FINAL}Random_assembly_ACC.txt # list of all the genomes it tried downloading
    downloadedGenomes=$(find ${GENOMES} -type f \( -iname \*.fna -o -iname \*.fna.gz \) -printf '%F\n' | perl -p -e 's/^(GCF_.*?)_.*/$1/' | sort)
    rm ${FINAL}More_assembly_ACC.txt

    # 02 Detect 16S in genomes
    ## run barrnap on each genome (for determining the number of 16S gene copies that could potentially be amplified)
    echo -e "\n02 Detect 16S rRNA genes in genomes using barrnap" >>${RUNLOG}
    cd $GENOMES
    for gz_genome in *.gz; do
      gunzip -f ${gz_genome}
      barrnap -k bac ${gz_genome%.gz} --reject 0.8 -t 10 > ../${RAW}/Genome_hits/${gz_genome%.gz}.barrnap 2>&1
    done
    rm -rf *.gz # for some reason some files retain their gz copy
    conda deactivate

```

```
    # # make sure there are no barrnap files without matching genome files (I'm not sure why this happens)
    # mismatch=$(comm -3 <(cat<<<$(find ../${RAW}/Genome_hits/ -type f -name "*.barrnap" |  perl -p -e 's/.*(GCF_.*?)_.*/$1/' | sort)) <(cat<<<$(find ./ -name "*.fna" | perl -p -e 's/.*(GCF_.*?)_.*/$1/' | sort)))
    # if [ ${#mismatch} -gt 0 ]; then
    #   cat <<< $mismatch | while read line; do
    #     echo -e "removing ../${RAW}/Genome_hits/${line}* \n" >>${RUNLOG}
    #     rm -f ../${RAW}/Genome_hits/${line}*
    #     echo -e "../${RAW}/Genome_hits/${line}* \n" >>${RUNLOG}
    #   done
    # fi

    # find and remove genomes without 16S (or 16S length < 0.8)
    nGenome16SHits=$(cat ../${RAW}/Genome_hits/*.barrnap | grep "Name=16S" | wc -l) # total 16S hits in all the genomes (incl. all paralogs)
    nGenomesw16S=$(find ../${RAW}/Genome_hits/ -type f -name "*.barrnap" -exec grep -m 1 "Name=16S" '{}' ';' | wc -l)
    echo -e "barrnap detected at least one 16S rRNA gene in $nGenomesw16S out of $(find ./ -name "*.fna" | wc -l) downloaded genomes" >>${RUNLOG}
    GenomeswoI6S=$(find ../${RAW}/Genome_hits/ -type f -name "*.barrnap" -exec grep -HL "Name=16S" '{}' ';' | perl -p -e 's/.*(GCF_.*?)_.*/$1/')
    if [ ${#GenomeswoI6S} -gt 0 ]; then
      cat <<< $GenomeswoI6S | while read line; do
        # echo -e ${line} >>${RUNLOG}
        rm -f ${line}* 2> /dev/null
        rm -f ../${RAW}/Genome_hits/${line}* 2> /dev/null
      done
    fi

    # run in-silico PCR on the genomes
    echo -e "Run in-silico PCR on each genome" >>${RUNLOG}
    #if [ ! -f PCR_genome_hits.tsv ]; then
        # echo -e  "#seq_name\tbarrnap_V1\tGene\tStart\tEnd\tevalue\tstrand\t?\tProduct\n" > PCR_genome_hits.tsv
        echo -e  "Seq name\tACC #\tF position\tF mismatches\tR position\tR mismatches\tAmplicon length" > PCR_genome_hits.tsv
    #fi
    for genome in *.fna; do
      primersearch ${genome} "../${OLIGOS}" 20 -scircular1 Y ${genome%.seq}.match # 10% mismatches from primer allowed
      AMPLIMERS=$(grep Amplimer\ length ${genome%.seq}.match | awk -F $' ' 'BEGIN {OFS = FS} $3<1600' | wc -l) # number of amplicons <1600 bp
      if [ ${AMPLIMERS} -gt 0 ]; then
        cat ${genome}.match \
        | awk -v fname=$(basename ${genome%.fna}) 'BEGIN{RS="Amplimer [0-9]+";FS="\n";}NR>1{printf fname"";for(i=2;i<=NF;i++){printf($i)};print("")}' \
        | awk 'BEGIN{FS="\t"}{$3=""; print}' \
        | awk 'BEGIN{FS=" ";OFS="\t"}{gsub(/[\[,\]]/,"",$18); print $1,$3,$9,$11,$18,$20,$24}1'\
        | awk -v GL="$GeneLen" 'BEGIN {OFS=FS} $7<GL' >> PCR_genome_hits.tsv
      fi
    done

    nGenomePCRHits=$(tail -n+2 PCR_genome_hits.tsv | wc -l) # total 16S-PCR hits in all the genomes (incl. all paralogs)
    nGenomeswPCR=$(awk 'NR==1 {next} !seen[$1]++' PCR_genome_hits.tsv | wc -l) # no. of genomes with at least one 16S PCR hit
    echo -e "primersearch detected at least one 16S rRNA amplicon in ${nGenomeswPCR} out of $(find ../${GENOMES} -name "*.fna" | wc -l) downloaded genomes" >>${RUNLOG}
    pAmplified=$(echo "nGenomePCRHits/$nGenome16SHits" | bc -l)
    echo -e "${nGenomePCRHits} 16S PCR hits were detected in the genomes out of ${nGenome16SHits} total 16S hits ($(printf '%.3f' ${pAmplified}))" >>${RUNLOG}

    # remove genomes with no 16S match
    goodGenomes=$(cat PCR_genome_hits.tsv | awk '!(NR<=1) {print $1}' | perl -p -e 's/^(GCF_.*?)_.*/$1/' | sort | uniq)
    badGenomes=$(comm -3 <(cat<<<$downloadedGenomes) <(cat<<<$goodGenomes))

    if [ ${#badGenomes} -gt 0 ]; then
      cat <<< $badGenomes | while read line; do
        # echo -e ${line} >>${RUNLOG}
        rm -f ${line}* 2> /dev/null
        rm -f ../${RAW}/Genome_hits/${line}* 2> /dev/null
      done
    fi
    cd ../
    GCOUNT=$(find ${GENOMES} -name "*.fna" | wc -l)
done # end while

fgrep -f <(cat <<<$downloadedGenomes) ${FINAL}/Random_assembly_ACC.txt > ${FINAL}Downloaded_genomes.txt # list of all downloaded genomes
echo -e "$(find ${GENOMES} -name "*.fna"  | wc -l) genomes were downloaded successfully" >>${RUNLOG}
cp ${GENOMES}PCR_genome_hits.tsv ${FINAL}

# Again make sure there are no barrnap files without matching genome files (I'm not sure why this happens)
# mismatch=$(comm -3 <(cat<<<$(find ${RAW}/Genome_hits/ -type f -name "*.barrnap" |  perl -p -e 's/.*(GCF_.*?)_.*/$1/' | sort)) <(cat<<<$(find ./${GENOMES} -name "*.fna" | perl -p -e 's/.*(GCF_.*?)_.*/$1/' | sort)))
# if [ ${#mismatch} -gt 0 ]; then
#   cat <<< $mismatch | while read line; do
#     echo -e "removing ${RAW}/Genome_hits/${line}* \n" >>${RUNLOG}
#     rm -f ${RAW}/Genome_hits/${line}*
#     echo -e "${RAW}/Genome_hits/${line}* \n" >>${RUNLOG}
#   done
# fi

nGenome16SHits=$(cat ${RAW}/Genome_hits/*.barrnap | grep "Name=16S" | wc -l) # total 16S hits in all the genomes (incl. all paralogs)
nGenomesw16S=$(find ${RAW}/Genome_hits/ -type f -name "*.barrnap" -exec grep -m 1 "Name=16S" '{}' ';' | wc -l)
echo -e "barrnap detected at least one 16S rRNA gene in $nGenomesw16S out of $(find ./${GENOMES} -name "*.fna" | wc -l) downloaded genomes" >>${RUNLOG}

# 03 Fragment genomes
conda activate SIPSim
# Generate a list of genome fragments based on a given distribution
echo -e "\n03 Fragment genomes" >>${RUNLOG}
# Awk create genome_index.txt (needed for SIPSim fragments)
find ${GENOMES}*.fna | awk 'sub( /.*\//,"",$0 )' | awk -v OFS='\t' '$1=$1"\t"$1' | awk -v OFS='\t' '{gsub(".fna$","",$1); print($1,$2)}' > ${GENOMES}genome_index.txt

# run SIPSim fragments
nFRAGS2grab=$(echo "scale=0; ($nFRAGS * 1.1)/1" | bc) # add 10% because Grab_frags.py fails on some fragments
# output to FRAGMENTS (skewed-normal params are empirically measured, see fit_selm.html)
SIPSim fragments  \
    $GENOMES/genome_index.txt \
    --fp $GENOMES \
    --f1d skewed-normal,2547,6631,5 \
    --flr None,None \
    --nf $nFRAGS2grab \
    --np=20 \
    --tbl \
    > ${FRAGMENTS}Frags_table.txt
cp ${FRAGMENTS}Frags_table.txt  ${FINAL}
conda deactivate
echo -e "SIPSim fragments generated $(tail -n+2 ${FRAGMENTS}Frags_table.txt | wc -l) DNA fragments (${nGENOMES} * ${nFRAGS} +10%)" >>${RUNLOG}

# Grab the fragments from the genomes using the table
echo -e "Grab the fragments from the genomes" >>${RUNLOG}
python Grab_frags.py $GENOMES $FRAGMENTS
dDownloadedFrags=$(find ${FRAGMENTS} -name "*.fa" | wc -l)
# skirts the unwanted fragments
totalFrags=$(echo "scale=0; (${nFRAGS} * ${nGENOMES})/1" | bc) # total desired fragmnes
if [ ${dDownloadedFrags} -gt ${totalFrags} ]; then
  tooManyFrags=$(echo "scale=0; (${dDownloadedFrags} - ${totalFrags})" | bc)
  find ${FRAGMENTS} -name "*.fa" | sort -R | head -n ${tooManyFrags} | xargs rm
fi

realnFRAGS=$(find ${FRAGMENTS} -name "*.fa" | wc -l)
echo -e "Grab_frags.py grabbed $realnFRAGS DNA fragments" >>${RUNLOG}

# plot the fragment distribution
echo -e "Plot the fragment distribution" >>${RUNLOG}
Rscript Plot_fragments.R
```

```
# 04 Detect 16S rRNA genes in the fragments
echo -e "\n04 Detect 16S rRNA genes in the fragments" >>${RUNLOG}
# run barrnap on each fragment
conda activate base
cd ${FRAGMENTS}
for fasta in *.fa; do
    #echo -e "$fasta"
    barrnap -k bac ${fasta} --outseq ../${RAW}/${fasta} -t 5 > ../${RAW}/${fasta%.fa}.barrnap 2>&1
done
conda deactivate

mkdir ../${FINAL}/Fragment_hits
## copy frags with 16S hits only
for rawResult in ../${RAW}/*.fa; do
  if [ -s $rawResult ] && $(cat $rawResult | grep -q 16S); then
    cp ${rawResult} ${rawResult%.fa}.barrnap ../${FINAL}/Fragment_hits
  fi
done
echo -e "barrnap detected at least one 16S rRNA gene in $(for HIT in ../${FINAL}/Fragment_hits/*.barrnap; do cat ${HIT} | grep -m 1 "Name=16S"; done | wc -l) out of $realnFRAGS fragments" >>${RUNLOG}

# Output all hits to files and summarise in a table
cd ../${FINAL}
cat ./Fragment_hits/*.barrnap | grep "Name=16S" | awk 'BEGIN{printf "#seq_name\tbarrnap_V\tGene\tStart\tEnd\tevalue\tstrand\t?\tProduct\n"} {print $0}' > All_hits.tsv

# 05 in silico PCR on fragments
echo -e "\n05 Run in-silico PCR on each fragment" >>${RUNLOG}
echo -e  "Seq name\tFragment\tF position\tF mismatches\tR position\tR mismatches\tAmplicon length" | tee -a PCR_hits.tsv Mult_PCR_hits.tsv
for HIT in ./Fragment_hits/*.fa; do
  primersearch ${HIT} "../${OLIGOS}" 20 -scircular1 N ${HIT%.seq}.match # 20% mismatches from primer allowed
  AMPLIMERS=`grep Amplimer\ length ${HIT%.seq}.match | wc -l`
  if [ ${AMPLIMERS} -gt 0 ]; then
    cat ${HIT}.match \
    | awk -v fname=$(basename ${HIT%.fna}) 'BEGIN{RS="Amplimer [0-9]+";FS="\n";};NR>1{printf fname"";for(i=2;i<=NF;i++){printf($i)};print("")}' \
    | awk 'BEGIN{FS="\t"}{$3=""; print}' \
    | awk 'BEGIN{FS=" ";OFS="\t"}{print $1,$3,$9,$11,$18,$20,$24}1' \
    | awk -v GL="$GeneLen" 'BEGIN {OFS=FS} $7<GL' >> PCR_hits.tsv
  fi
done

# 06 Summarise
echo -e "\n06 Summarise results" >>${RUNLOG}
# fragments with 16S
mean16SperGenome=$(echo -e "$(cat PCR_genome_hits.tsv | wc -l)/${nGENOMES}" | bc -l)
echo -e "Average number of 16S genes per genome: $(printf '%.1f' ${mean16SperGenome})" >>${RUNLOG}
nHITs=$(echo -e "$(find ./ -name "*.fa" | wc -l)" | bc -l)  # count frags with (one or more) 16S  (i.e. once each hit fragment)
echo -e "${nHITs} frags with (one or more) 16S gene copies (i.e. once each hit fragment)" >>${RUNLOG}
pHITs=$(echo -e "${nHITs}/${realnFRAGS}" | bc -l)  # fraction of frags with (one or more) 16S per genome per fragment (i.e. once each hit fragment)
echo -e "Fraction of frags with (one or more) 16S per genome per fragment (i.e. once each hit fragment): $(printf '%.4f' ${pHITs})" >>${RUNLOG}
nMultHits=$(awk 'c[$1]++ && c[$1]==2' All_hits.tsv | wc -l) # count fragments with more than 1 16S hit (i.e. >1 coord lines)
echo -e "${nMultHits} fragments with more than 1 16S hit (i.e. >1 coord lines)" >>${RUNLOG}
pMultHits=$(echo "${nMultHits}/${nHITs}" | bc -l) # fraction of fragments with more than 1 16S hit (i.e. >1 coord lines) per fragments with 16S
echo -e "Fraction of fragments with more than 1 16S hit (i.e. >1 coord lines) per fragments with 16S: $(printf '%.3f' ${pMultHits})" >>${RUNLOG}
# fragments with PCR amplicon
awk 'NR==FNR {count[$1]++; next} count[$1]>1' PCR_hits.tsv PCR_hits.tsv >> Mult_PCR_hits.tsv # grab the fragments with more than one PCR hit per fragment
redundantHits=$(awk 'c[$1]++; c[$1]==2' PCR_hits.tsv | awk 'seen[$1]++' | wc -l) # count redundant hits (# multiple hits excl. first occurrence)
nPCRHits=$(echo -e "($(tail -n +2 PCR_hits.tsv | wc -l)-${redundantHits})" | bc -l) # count ddPCR hits (i.e only one hit per fragment is counted)
echo -e "No. of gene copies identified by ddPCR (PCR hits - hits on the same fragment): ${nPCRHits}" >>${RUNLOG}
pPCRHits=$(echo -e "${nPCRHits}/${nHITs}" | bc -l) # fraction of ddPCR hits per fragments with 16S
echo -e "Fraction of ddPCR hits per fragments with 16S (i.e. fraction of non-split genes): $(printf '%.2f' ${pPCRHits})" >>${RUNLOG}
nMultPCRHits=$(awk 'c[$1]++ && c[$1]==2' Mult_PCR_hits.tsv | wc -l) # count fragments multiple PCR hits (i.e. once per fragment!)
echo -e "${nMultPCRHits} fragments contain more than one 16S gene copy" >>${RUNLOG}
pMultPCRHits=$(echo -e "${nMultPCRHits}/${nPCRHits}" | bc -l) # fraction of multiple PCR hits per fragments with PCR hit
echo -e "Fraction of multiple PCR hits per fragments with a PCR hit: $(printf '%.2f' ${pMultPCRHits})" >>${RUNLOG}
pFragWPCRFragmentsW16S=$(echo -e "${nPCRHits}/${nHITs}" | bc -l) # fraction of PCR hits in total number of fragments with 16S (a measure of how many genes were split)
echo -e "Fraction of of PCR hits in total number of fragments with 16S (a measure of how many genes were split): $(printf '%.2f' ${pFragWPCRFragmentsW16S})" >>${RUNLOG}
redundant16S=$(awk 'c[$1]++; c[$1]==2' PCR_genome_hits.tsv | awk 'seen[$1]++' | wc -l) # count redundant 16S genes in genomes (# multiple 16S excl. first occurrence)
nGenomesWPCR=$(echo -e "($(tail -n +2 PCR_genome_hits.tsv | wc -l)-${redundant16S})" | bc -l) # count ddPCR hits for whole genomes (i.e only one hit per fragment is counted) if not = $nGENOMES then there were PCR mismatches
echo -e "${nGenomesWPCR} ddPCR hits for whole genomes (i.e only one hit per fragment is counted). If not =nGENOMES then there were PCR mismatches" >>${RUNLOG}

# Correct ddPCR value
ddPCRcFactor=$(echo -e "(1/${pFragWPCRFragmentsW16S} + ${redundantHits}/${nPCRHits})" | bc -l) # correct for fraction of split genes + for multiple amplicons per fragment
cddPCR=$(echo -e "${nPCRHits}*(${ddPCRcFactor})" | bc -l)

# Create a summery table
echo -e "# Genomes,# Frags per genome,Total # frags,Mean 16S per genome,Frags w. 16S,P frags w. 16S,Frags w. PCR (ddPCR outcome), P frags w. PCR,Frags w. multi 16S,P frags w. multi 16S,Frags w. multi PCR,P frags w. multi PCR,ddPCR correction factor, Corrected ddPCR value" \
> Summary.csv
echo -e "${nGENOMES},`printf '%.0f' $(echo "$realnFRAGS / $nGENOMES" | bc -l)`,$(echo "$nGENOMES * $realnFRAGS" | bc -l),`printf '%.1f' ${mean16SperGenome}`,`printf '%.0f' ${nHITs}`,`printf '%.4f' ${pHITs}`, `printf '%.0f' ${nPCRHits}`,`printf '%.4f' ${pPCRHits}`, \
`printf '%.0f' ${nMultHits}`,`printf '%.4f' ${pMultHits}`,`printf '%.0f' ${nMultPCRHits}`,`printf '%.4f' ${pMultPCRHits}`,`printf '%.3f' ${ddPCRcFactor}`,`printf '%.0f' ${cddPCR}`" >> Summary.csv
if [ $(cat Summary.csv | wc -l | bc -l) == 2 ]; then
    echo -e "Results written to file successfully" >>${RUNLOG}
fi

cd ../
mv ${RUNLOG} ${FINAL}
rm -rf Fragments
zip -rm Genomes.zip Genomes
rm -rf ${RAW}

# [1] https://github.com/kblin/ncbi-genome-download
# [2] Seemann T. barrnap 0.9 : rapid ribosomal RNA prediction, https://github.com/tseemann/barrnap
# [3] SIPSim

```

10. R Script

```r
# Plot genome fragment distribution
library(ggplot2) # ggplot aes geom_point geom_density2d scale_color_discrete labs theme_bw theme element_text guides guide_legend ggsave
library(readr) # read_tsv
library(stringr) # str_replace %>%
library(dplyr) # %>% pull filter
library(ggExtra) # ggMarginal
library(ragg) # scaling option in ggsave

# wd <- "/proj/Methods_dev/ddPCR/SIPSim_example/Fragments/"
wd <- "./Fragments/"
out_dir <- "./Results/"
Fragments <- read_tsv(paste0(wd, "Frags_table.txt"))
Taxa <- read_tsv(paste0(out_dir, "Random_assembly_ACC.txt"), col_names = FALSE)
Fragments$Taxon <- Taxa$X2[match(str_replace(Fragments$taxon_name,
                                             "(^.*?_.*?)_.*", "\\1"),
                                 Taxa$X1)]

# retain no more than 100 random taxa
if (length(unique(Fragments$taxon_name)) > 100) {
  Fragments %>%
    pull(taxon_name) %>%
    unique() %>%
    sample(100) -> subset
  Fragments <- filter(Fragments, taxon_name %in% subset)
}

p <- ggplot(Fragments, aes(fragGC, fragLength, color = Taxon, group = taxon_name)) +
  geom_point(alpha = 0) +
  geom_density2d(alpha = 1/5) +
  scale_color_discrete('Taxon') +
  labs(x = 'Fragment G+C (%)', y = 'Fragment length (bp)') +
  theme_bw() +
  theme(text = element_text(size = 14),
        legend.text = element_text(margin = margin(r = 1, unit = 'cm')))  + # space out the legend
  guides(colour = guide_legend(override.aes = list(alpha = 1/2),
         nrow = length(unique(Fragments$Taxon))/3)) +
  # guides(colour = guide_legend(override.aes = list(alpha = 1/2))) +
  theme(legend.position = "bottom")
p <- ggMarginal(p, groupColour = TRUE, groupFill = TRUE, alpha = 1/5)

# delete a file created (prob.) by ggMarginal
file.remove("Rplots.pdf")

pngfile <- fs::path(out_dir,  "Frag_dist.png")
ggsave(
  pngfile,
  p,
  device = agg_png,
  width = 22,
  height = 22,
  units = "cm",
  res = 300,
  scaling = 0.9
)
```