Record-seq Pipeline Development

Lab rotation report

Wiona Sophie Glänzer

12.09.-21.10.2022

Advisors:
Tanmay Tanna
Randall J. Platt

D-BSSE
ETH Zürich

Abstract

Record-seq is a recently developed technology that enables bacterial cells to record their own transcriptional activity into DNA and preserve it over time. Record-seq readout consists of deep sequencing and data interpretation using a dedicated computational analysis pipeline. This report describes improvements to and experiments with the Record-seq data analysis pipeline. First, we worked on the primary analysis pipeline, augmenting its capability to detect recorded sequences from reads using updated approximate string matching methods. Next, we tested several methods for normalizing counts distributions during secondary analysis to apply differential expression analysis tools developed for RNA sequencing. Two new normalization methods, namely plasmid normalization, and percentile normalization were implemented.

Contents

1 Transcriptional recording with RecordSeq

RNA sequencing measures the abundance of transcripts within cells and thus offers insight into cellular states and processes. Based on the abundance of observed transcripts, we can also draw conclusions about the environment that the observed cells are inhabiting. However, RNA-seq is inherently disruptive and is limited to showing us only one snapshot corresponding to the time of RNA extraction, whereas environmental and cellular states are dynamically changing.

In order to learn about previous states of a cell and elucidate the history of its response to the environment, cellular recording techniques have been developed [1]. Record-seq uses CRISPR spacer acquisition to insert snippets from RNA transcripts into a plasmid, which gets passed on to daughter cells and thus allows us to measure past transcript abundances even days later [2]. Record-seq uses a *Fusicatenibacter saccharivorans* RT-Cas1–Cas2 (*Fs*RT-Cas1–Cas2) complex for spacer acquisition. Since spacer acquisition is a rare event, a selective amplification method called SENECA is used prior to sequencing to get an optimal readout [3].

The data used in this project comes from recent experiments in which Record-seq was deployed in *E. coli* cells, which were then used as sentinel cells in the murine gut [4]. Specifically, the mice were divided into three groups and fed different diets for seven days, following which they were switched to the same diet. Fecal Record-seq samples were collected over 20 days, with matching fecal RNA-seq samples being collected on days 7, 9, 12, 14 and 20. The experimental design is shown in figure 1.

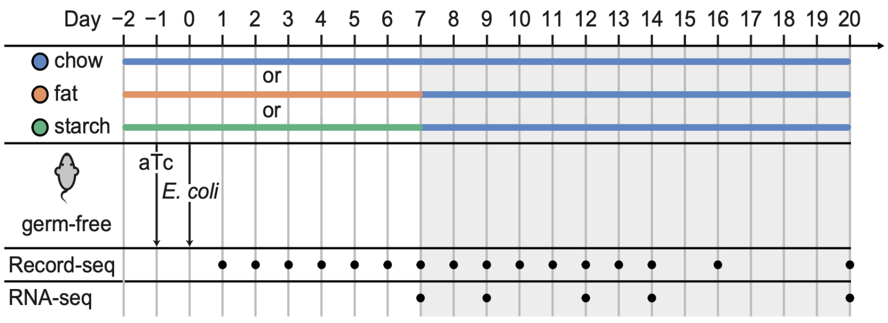

Day −2 −1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
chow
fat
starch
or
or
aTc
E. coli
germ-free
Record-seq
RNA-seq

Figure 1: Experimental design, figure 1D reproduced from [4]

2 The Record-seq Pipeline

To analyze the experimental results, the obtained sequencing reads are first processed using the primary analysis pipeline, which consists of several python scripts and programs combined in one Snakemake [5] workflow. Secondary analysis is performed using the recoRdseq package in R.
The snakemake workflow consists of the following steps:

1. Run FASTQC [6] and trimmomatic [7] for quality control and adapter trimming
2. Convert FASTQ files to FASTA files
3. Extract spacers from reads using *spacerExtractor.py*
4. Align spacers to the genome and plasmid reference using bowtie2 [8], process alignments using samtools and deduplicate alignments using *SErmdup.py*
5. Generate counts matrices using featurecounts [9]
6. Count normalization
7. Statistical analysis including differential gene expression

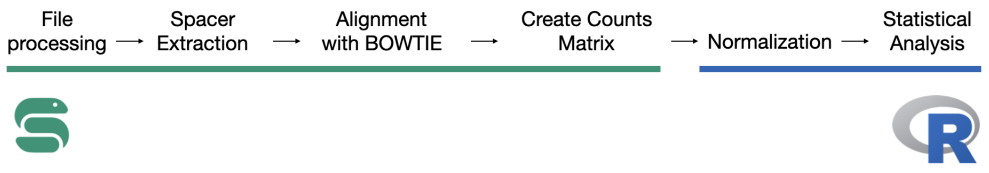


Figure 2: Steps in the Record-seq analysis pipeline

In this project, we focused on incrementally improving the spacer extraction approach (*spacerExtractor.py*) and introducing and comparing new count normalization methods.

3 Spacer Extraction

Given the sequencing reads, the next step is to identify and extract the acquired spacers. To do so, the direct repeat sequences of the CRISPR array, which delimit the spacers, need to be identified. As shown in figure 3, there can either be one acquired spacer delimited by a direct repeat 1 and a direct repeat 2 or two spacers separated by a full direct repeat. Acquisition of more than two spacers is also possible and a goal for further development of Record-seq; however, it is currently very rare and not included in the computational analysis.

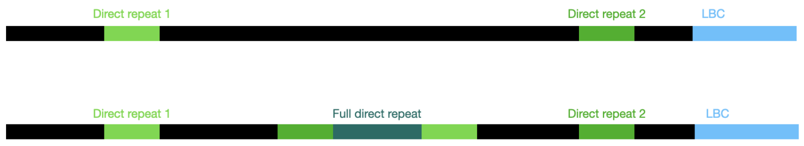


Figure 3: Read with direct repeats and acquired spacers, top: single acquisition, bottom: double acquisition

3.1 Number of mismatches

In initial Record-seq experiments, there were a high number of mismatches in the direct repeat sequences. These mismatches could have been introduced as a result of amplification errors, sequencing errors or errors during acquisition/replication of the recorded sequence. To account for these mismatches, a fuzzy search based approach, which finds matches that are up to a certain Levenshtein distance different from the repeat sequence, was implemented for finding the direct repeats.

Definition 3.1. *The **Levenshtein distance** or **Edit distance** between two strings a and b is defined as the number of edits (insertions, deletions, and character changes) that are necessary to transform a into b.*

To identify whether improvements in the experimental approach for Record-seq have influenced the number of mismatches in the repeat sequences and whether the fuzzy search based approach is still necessary, we counted the number of repeat sequences and library barcodes with mismatches in current experimental data. Overall each of these sequence elements was found as a perfect match in 68%, with Levenshtein distance 1 in 11% and with distance 2 in 2% of the $166,234,522$ reads. These numbers are skewed by the full direct repeat which occurs only in 0.1% of reads as a perfect match, while direct repeat one appears in 95%, direct repeat two in 91% and the library barcode in 88%.Owing to the high number of sequence elements found with mismatches, we concluded that the fuzzy search is a necessary part of the spacer extractor.

We counted that 91.2% of mismatches were substitutions and 8.8% were insertions or deletions. These percentages were approximately the same for all types of sequence elements, but there were more indels among the full direct repeat and the library barcode.

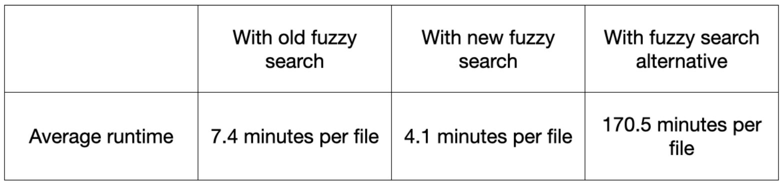
3.2 A new fuzzysearch

After an update of the fuzzysearch package [10], find_near_matches(), which was used as the main string matching function for sequences with mismatches, had a different return value format. As a result, the spacer extractor script only worked with an older version of the package. We updated the spacer extractor to work with the newest version of this package. We observed that this update improved performance and reduced runtime.
However, as backward compatibility of the package is not guaranteed, we also implemented another approach for approximate string matching that is specific to our problem. The pseudocode below shows the algorithm:

Algorithm 1 Fuzzysearch replacement function

```
Require: read, sequence to be found, maximal distance $d$
  fuzzyfile ← $a$ s.t. Levenshtein($a$, sequence) $< d$ as list sorted by distance
  for $b$ in fuzzyfile do
      if $b$ is in read then return position of $b$
      end if
  end for
```

The "fuzzyfiles" containing sorted lists of sequences with all possible mismatches only need to be generated once, and the runtime of the direct search step is $\mathcal{O}(n)$, where n is the length of the shorter of the two sequences being matched. In the classical fuzzy search approach, the equivalent step is the Levenshtein distance-based search, which has runtime $\mathcal{O}(dn)$, where d is the maximal allowed distance and n is again the length of the shorter sequence.
This means that in theory, the new algorithm could achieve a better runtime. When testing the new algorithm, however, we noticed a much longer runtime in comparison to using the fuzzysearch package (see Figure 4). This is likely due to algorithmic optimizations and the implementation of the fuzzysearch package in C++, as opposed to our function, which was implemented in Python.
To speed up our algorithm, we considered reducing the number of times the search function needs to be called by reusing results from previous runs. The full repeat consists of a direct repeat 1 and a direct repeat 2 with a short additional sequence in between, such that the positions of direct repeats could be used to find the full repeat. As a further step, a dynamic programming algorithm specific to searching for several similar sequence elements could be developed.



	With old fuzzy search	With new fuzzy search	With fuzzy search alternative
Average runtime	7.4 minutes per file	4.1 minutes per file	170.5 minutes per file

Figure 4: To compare the runtimes of the different versions of the spacer extractor, we tested it on six representative FASTA-files with a total of 18,666,366 reads. This table shows the average runtime per file.

4 Secondary analysis and normalization

In RNA sequencing analysis, it is typically assumed that the transcript counts for all samples and sample groups come from the same distribution i.e. transcript counts should proportionally represent how strongly expressed the gene is at the time of RNA extraction. In Record-seq, on the other hand, not only how highly expressed a gene is, but also how often it is acquired as a spacer influences the corresponding gene-aligning counts. We still assume that the rate of acquisition is the same for any gene within the same sample, but there are differences in acquisition rate across samples depending on the conditions of the sample, such as environment of the sentinel cell, biomass, overall transcription and other factors.

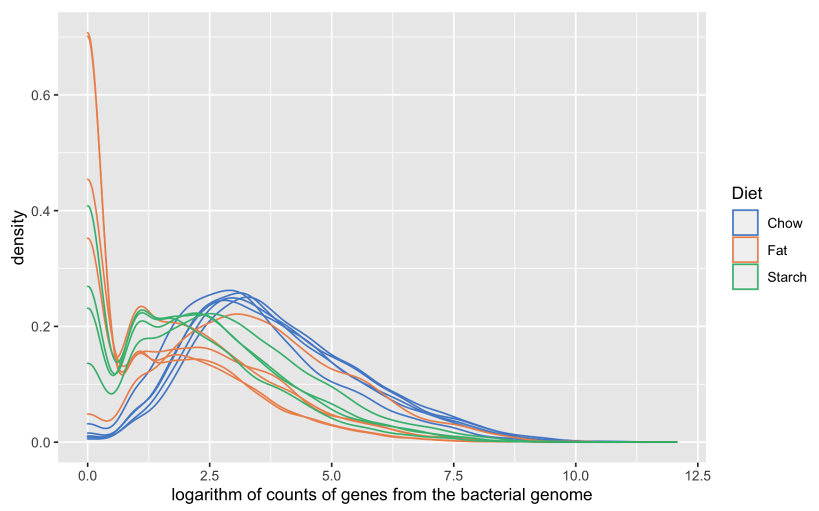


Figure 5: Density plot of the logarithm of counts of genes from the bacterial genome in samples collected on day 7, each line represents one sample

Figure 5 illustrates the different counts distributions across diet groups on day 7 in the experiment described in Figure 1. The fat and starch groups have zero-enriched distributions compared to the chow group. To compensate for these differences, an additional normalization of the counts is applied. The R packages DESeq2 [11] and edgeR [12] offer several standard normalization methods as built-in functions:

Definition 4.1. ***TMM:*** *This uses a trimmed mean of M-values, which are genes-wise log-fold expression changes. [13]*

Definition 4.2. ***VST:*** *The variance stabilizing transform makes samples approximately homoscedastic i.e. the variance independent from the mean [14].*

Definition 4.3. ***Size Factor Norm:*** *Normalization factors are calculated using a median-of-ratios method [11].*

Definition 4.4. ***thinCounts:*** *The thinCounts normalization uses a downsampling approach in which the sums of counts for each sample are scaled down to the minimum sum of counts using multinomial thinning [15].*

DESeq2 also offers a probabilistic version of thinCounts, but the RecoRdseq package uses the non-probabilistic version.

We compared these four approaches for normalizing the counts distributions across the diet groups (Figure 6). This analysis showed that the ”thincounts” or downsampling method was the most effective at rendering the distributions similar. However, this method involves removing observed counts and thus results in a loss of information; therefore, it is desirable to find an alternative approach. So far, vst normalization has been used for data analysis [4]. The following sections describe two new normalization methods that were developed and tested.

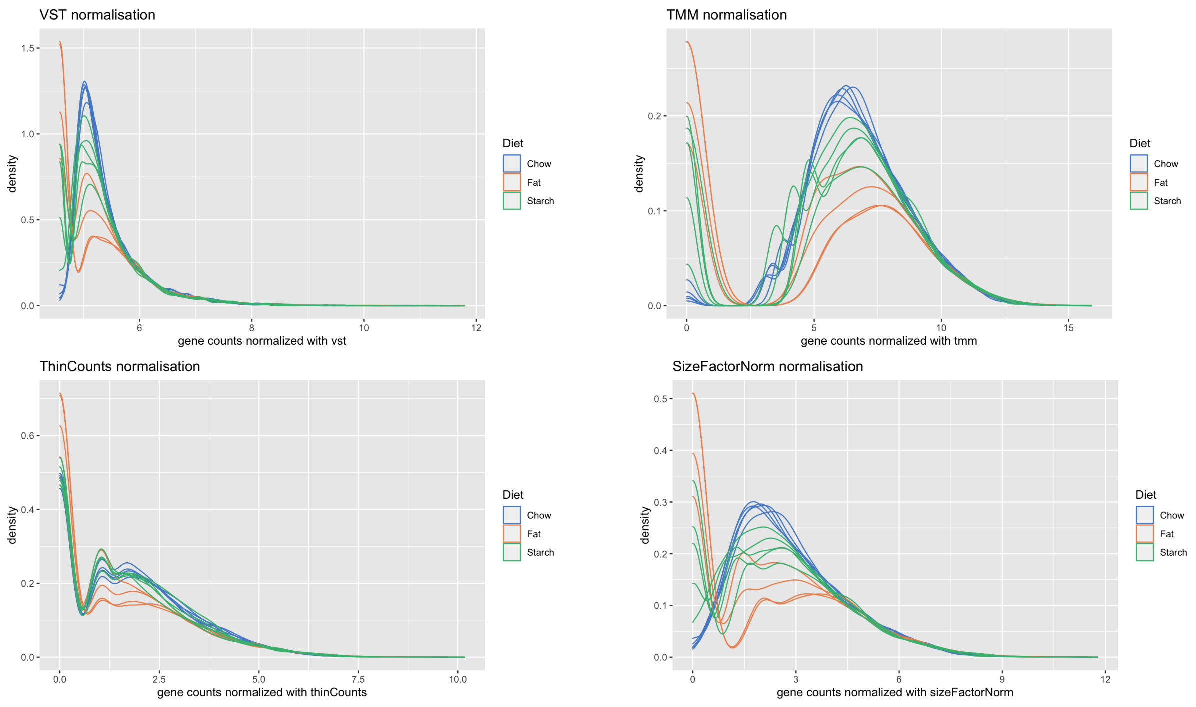


Figure 6: Counts densities for different normalizations

4.1 Plasmid normalization

In the **Plasmid normalization** method, we propose normalizing genomic gene counts using the counts corresponding to genes from the plasmid. The underlying assumption is that the expression of these genes is driven by synthetic promoters and is thus invariant to the treatment group, except in cases where the treatment causes global changes in transcription or acquisition efficiency. This is analogous to using "housekeeping genes" for normalizing transcript counts in eukaryotic transcriptomic data analysis.

On the plasmid, there are four genes: KanR, ROP, tetR, FsRT-Cas1, and FsRT-Cas2. The Cas1 and Cas2 genes are particularly suitable for normalization because the expression of the RT-Cas1-Cas2 complex directly influences the rate of spacer acquisition.
To decide on a specific normalization function using these two gene counts, we initially looked at the correlation between Cas1 and Cas2 counts (see Figure 7)

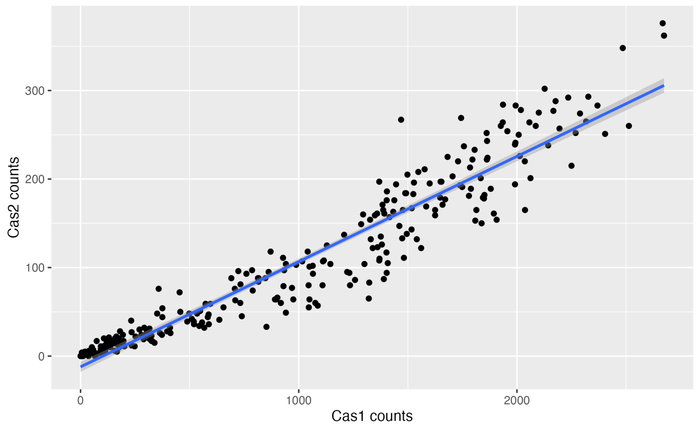


Figure 7: Correlation between Cas1 and Cas2, $R = 0.95, p < 2.2e - 16$

As Cas1 and Cas2 counts are highly correlated, we expect both to perform similarly for normalization. Nevertheless, we used the mean of the two counts as the normalization factor to reduce noise. Figure 8 shows the resulting counts distributions.

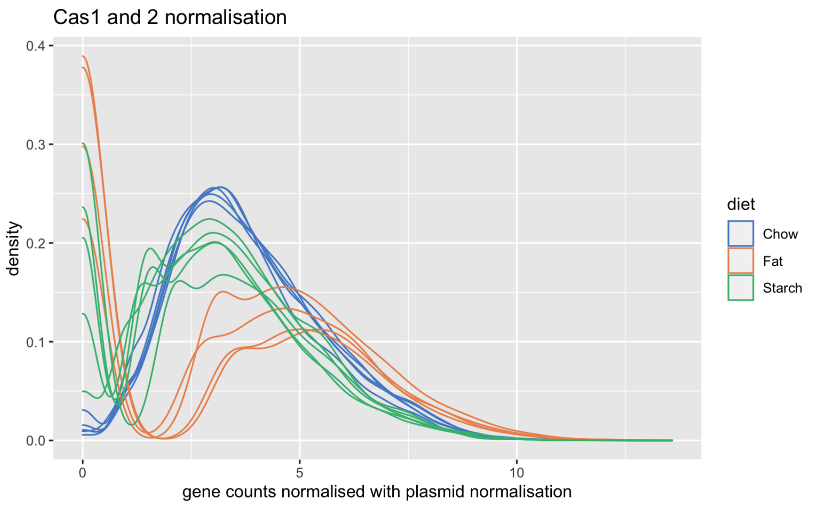
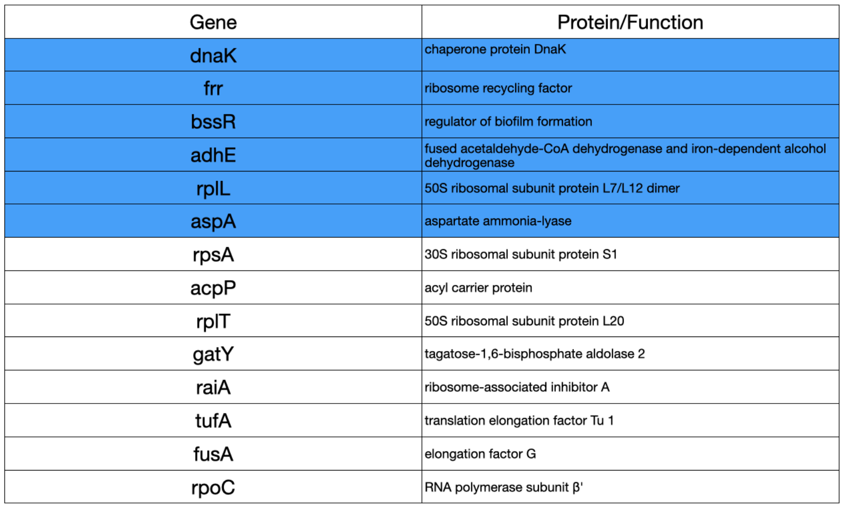


Figure 8: Counts distribution after plasmid normalization

4.2 Percentile normalization

Definition 4.5. *In* ***Percentile normalization****, the counts of a gene or several genes that are always highly expressed are used as a normalization factor.*

Here we assume that a measure of central tendency, such as the mean or median, for highly expressed genes, should be similar across samples irrespective of treatment group, and this measure can thus be used as a normalizing factor. In this report, we used all genes above the 99th percentile for normalization. To apply this method to the given data, we first identified genes whose expression was above the 99th percentile in each sample. Only considering samples from day 7, there were 14 such genes. Considering all samples, we found 6 such genes.



Gene	Protein/Function
dnaK	chaperone protein DnaK
frr	ribosome recycling factor
bssR	regulator of biofilm formation
adhE	fused acetaldehyde-CoA dehydrogenase and iron-dependent alcohol dehydrogenase
rplL	50S ribosomal subunit protein L7/L12 dimer
aspA	aspartate ammonia-lyase
rpsA	30S ribosomal subunit protein S1
acpP	acyl carrier protein
rplT	50S ribosomal subunit protein L20
gatY	tagatose-1,6-bisphosphate aldolase 2
raiA	ribosome-associated inhibitor A
tufA	translation elongation factor Tu 1
fusA	elongation factor G
rpoC	RNA polymerase subunit β'

Figure 9: Genes with an expression of at least the 99th percentile on day 7, in blue on all days, gene functions from [16]

As seen in figure 10, the counts distributions look more similar after quantile normalization than without normalization; however, significant differences can still be observed, particularly between the chow and the fat distributions.

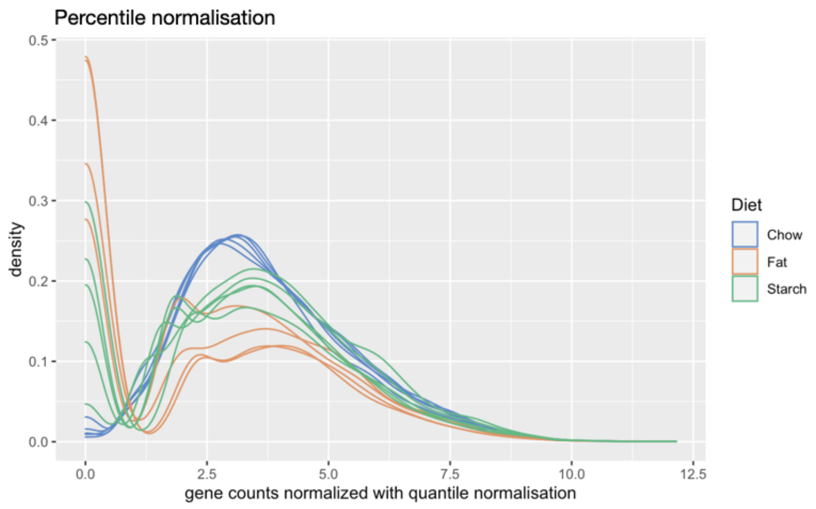


Figure 10: Counts distribution after percentile normalization

To optimize the percentile normalization, it would be a good idea to pick genes that one does not expect to be differentially expressed in the different groups. Genes that have been used for normalization must either be excluded from the downstream differential expression analysis or a high number of genes must be used for normalization. To increase the number of available genes, one could for example choose a different percentile above which all genes are used.

4.3 Quantitative comparison

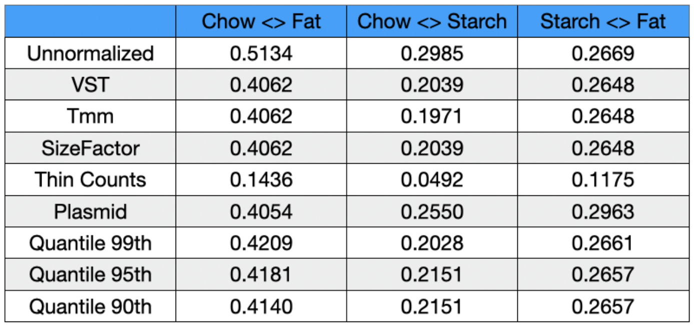
To compare the presented normalisation methods quantitatively, we calculated the Kolmogorov-Smirnov distances between the normalized distributions.

Definition 4.6. ***Kolmogorov-Smirnov statistic*** *When comparing a distribution with cumulative distribution function (CDF) F_n to an underlying distribution with CDF F_0, we define the KS-statistic as*

$$d_n = ||F_n - F_0|| = sup_x ||F_n(x) - F_0(x)||$$

With $sup_x(f(x))$ defined as the least upper bound on $f(x)$ when varying x, which in this case is the least upper bound on the set of distances between the two CDFs.

A table of all average values of the Kolmogorov-Smirnov statistic for the between-group comparisons are shown in figure 11 and corresponding matrices for each sample are shown as heatmaps in figure 12. All discussed normalization methods result in lower KS-values than for the unnormalized samples. As expected, the values are smallest for within-group comparisons for all normalizations. The highest values are found in the comparison of the fat group with the chow group, followed by the fat with starch comparison. The fat group also has the most zero-enriched original count distribution, and it is known that the conditions in the gut created by a high-fat diet reduce the *E.coli* biomass. As the newly implemented normalization methods perform better for some group comparisons, they might be useful in specific contexts, but they do not provide a complete solution to the normalization problem.



	Chow <> Fat	Chow <> Starch	Starch <> Fat
Unnormalized	0.5134	0.2985	0.2669
VST	0.4062	0.2039	0.2648
Tmm	0.4062	0.1971	0.2648
SizeFactor	0.4062	0.2039	0.2648
Thin Counts	0.1436	0.0492	0.1175
Plasmid	0.4054	0.2550	0.2963
Quantile 99th	0.4209	0.2028	0.2661
Quantile 95th	0.4181	0.2151	0.2657
Quantile 90th	0.4140	0.2151	0.2657

Figure 11: Average Kolmogorov-Smirnov statistic values

5 Conclusion

In the first part of the project, we worked on the Spacer Extractor script. It now works with the newest version of the fuzzysearch package and is almost twice as fast compared to the older version. The package-independent version is currently an order of magnitude slower and would need to be optimized with dynamic programming and coded in C++, to be production ready.
The tests of the different normalization methods showed that thincounts normalization equalizes the distributions under different conditions the most. Among the other tested methods, there were no strong differences. An alternative to normalizing the counts prior to differential expression analysis would be to use a rank-based method for differential analysis instead. Such methods are described for example in [17].

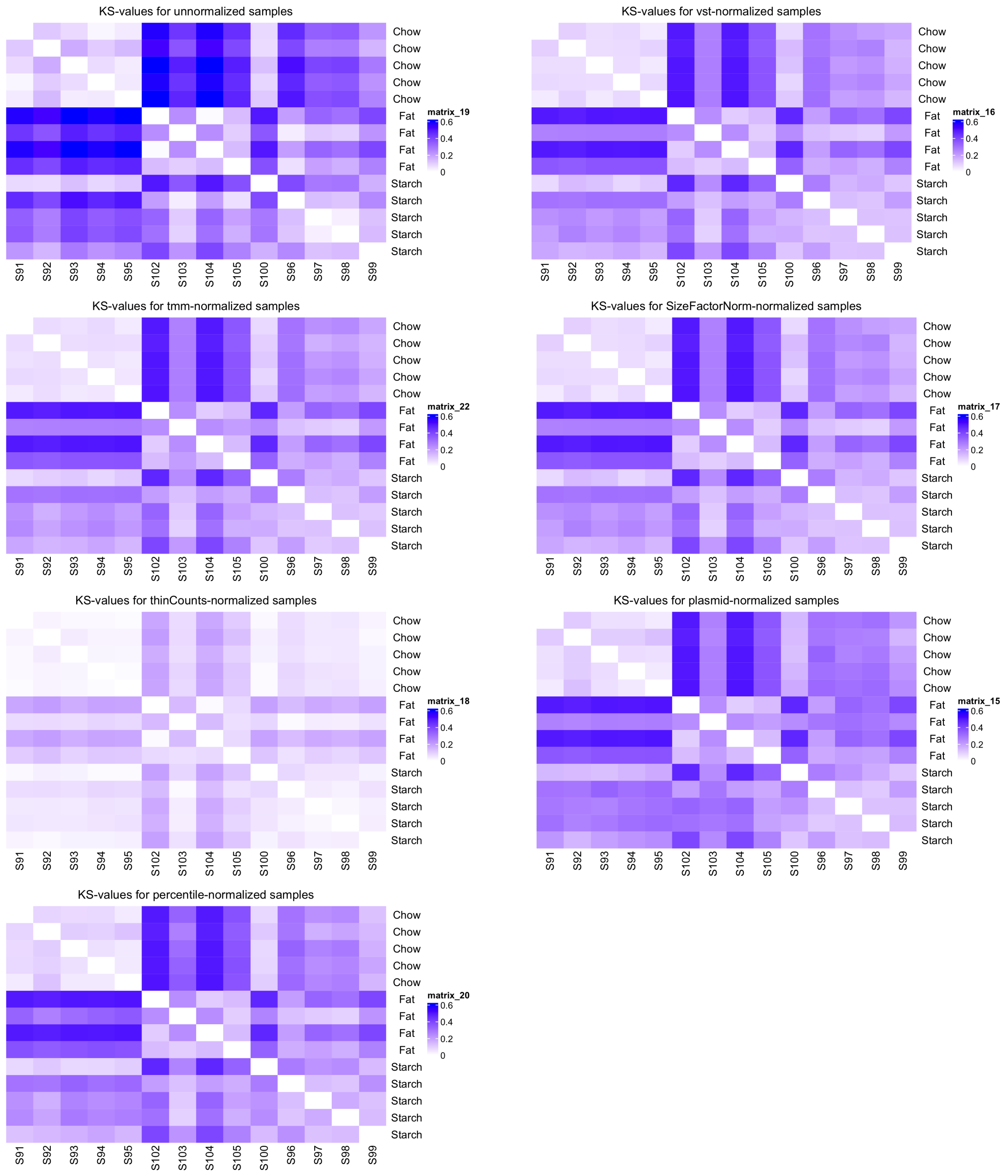


Figure 12: Kolmogorov-Smirnov test matrices

References

[1] Ravi U. Sheth and Harris H. Wang. “DNA-based memory devices for recording cellular events”. en. In: *Nature Reviews Genetics* 19.11 (Nov. 2018). Number: 11 Publisher: Nature Publishing Group, pp. 718–732. ISSN: 1471-0064. DOI: 10.1038/s41576-018-0052-8. URL: https://www.nature.com/articles/s41576-018-0052-8 (visited on 12/19/2022).

[2] Florian Schmidt, Mariia Y. Cherepkova, and Randall J. Platt. “Transcriptional recording by CRISPR spacer acquisition from RNA”. en. In: *Nature* 562.7727 (Oct. 2018). Number: 7727 Publisher: Nature Publishing Group, pp. 380–385. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0569-1. URL: https://www.nature.com/articles/s41586-018-0569-1 (visited on 12/19/2022).

[3] Tanmay Tanna et al. “Recording transcriptional histories using Record-seq”. en. In: *Nature Protocols* 15.2 (Feb. 2020). Number: 2 Publisher: Nature Publishing Group, pp. 513–539. ISSN: 1750-2799. DOI: 10.1038/s41596-019-0253-4. URL: https://www.nature.com/articles/s41596-019-0253-4 (visited on 12/19/2022).

[4] Florian Schmidt et al. “Noninvasive assessment of gut function using transcriptional recording sentinel cells”. In: *Science* 376.6594 (2022), eabm6038. DOI: 10.1126/science.abm6038. eprint: https://www.science.org/doi/pdf/10.1126/science.abm6038. URL: https://www.science.org/doi/abs/10.1126/science.abm6038.

[5] Felix Mölder et al. *Sustainable data analysis with Snakemake*. en. Tech. rep. 10:33. Type: article. F1000Research, Jan. 2021. DOI: 10.12688/f1000research.29032.1. URL: https://f1000research.com/articles/10-33 (visited on 12/19/2022).

[6] *Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence Data*. URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (visited on 11/19/2022).

[7] Anthony M. Bolger, Marc Lohse, and Bjoern Usadel. “Trimmomatic: a flexible trimmer for Illumina sequence data”. In: *Bioinformatics* 30.15 (Aug. 2014), pp. 2114–2120. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btu170. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4103590/ (visited on 12/19/2022).

[8] Ben Langmead and Steven L Salzberg. “Fast gapped-read alignment with Bowtie 2”. In: *Nature methods* 9.4 (Mar. 2012), pp. 357–359. ISSN: 1548-7091. DOI: 10.1038/nmeth.1923. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3322381/ (visited on 12/19/2022).

[9] Yang Liao, Gordon K. Smyth, and Wei Shi. “featureCounts: an efficient general purpose program for assigning sequence reads to genomic features”. eng. In: *Bioinformatics (Oxford, England)* 30.7 (Apr. 2014), pp. 923–930. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btt656.

[10] *fuzzysearch 0.3.0 documentation*. URL: https://fuzzysearch.readthedocs.io/en/latest/readme.html (visited on 11/19/2022).

[11] Michael I. Love, Wolfgang Huber, and Simon Anders. “Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2”. In: *Genome Biology* 15.12 (Dec. 2014), p. 550. ISSN: 1474-760X. DOI: 10.1186/s13059-014-0550-8. URL: https://doi.org/10.1186/s13059-014-0550-8 (visited on 12/19/2022).

[12] Mark D. Robinson, Davis J. McCarthy, and Gordon K. Smyth. “edgeR: a Bioconductor package for differential expression analysis of digital gene expression data”. In: *Bioinformatics* 26.1 (Jan. 2010), pp. 139–140. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp616. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2796818/ (visited on 12/19/2022).

[13] Mark D. Robinson and Alicia Oshlack. “A scaling normalization method for differential expression analysis of RNA-seq data”. In: *Genome Biology* 11.3 (Mar. 2010), R25. ISSN: 1474-760X. DOI: 10.1186/gb-2010-11-3-r25. URL: https://doi.org/10.1186/gb-2010-11-3-r25 (visited on 12/19/2022).

[14] *Variance Stabilizing Transformation: Apply a variance stabilizing transformation (VST) to the... in DESeq2: Differential gene expression analysis based on the negative binomial distribution*. en. URL: https://rdrr.io/bioc/DESeq2/man/varianceStabilizingTransformation.html (visited on 11/19/2022).

[15] *thinCounts: Binomial or Multinomial Thinning of Counts in hiraksarkar/edgeR_fork: Empirical Analysis of Digital Gene Expression Data in R*. en. URL: https://rdrr.io/github/hiraksarkar/edgeR_fork/man/thinCounts.html (visited on 12/24/2022).

[16] *EcoCyc: Encyclopedia of E. coli Genes and Metabolism*. URL: https://ecocyc.org/ (visited on 12/24/2022).

[17] Xiangyu Li et al. "A rank-based algorithm of differential expression analysis for small cell line data with statistical control". eng. In: *Briefings in Bioinformatics* 20.2 (Mar. 2019), pp. 482–491. ISSN: 1477-4054. DOI: 10.1093/bib/bbx135.