Supplementary Material for *Polyester*: simulating RNA-seq datasets with differential transcript expression

Alyssa C. Frazee, Andrew E. Jaffe, Ben Langmead, and Jeffrey T. Leek

March 11, 2015

1 GC Content Expression Models

Supplementary Figure 1 illustrates the 7 GC content expression models that ship with *Polyester*.

2 Positional Bias Models

For reference, we reproduce Supplementary Figure 3 of [4] here.

3 Error Models

Supplementary Figures 1-5 illustrate the error models available in *Polyester*, other than the one derived from TruSeq SBS Kit v5-GA chemistry, Illumina Genome Analyzer II, mate 1 (shown in Figure 2 in the main manuscript). In each of Supplementary Figures 1-5, separate panels are shown for each possible true reference nucleotide. Each panel illustrates the probability (y-axis) of mis-sequencing that reference nucleotide in a given read position (x-axis) as any of the 3 other nucleotides, or as an "N" (indicating an "unknown" nucleotide in the read). As expected, error probabilities increase toward the end of the read, and mate 2 error probabilities are higher than mate 1. The error models for Illumina Sequencing Kit v4 show some interesting spikes at certain read positions, which were also observed in the original analysis of this data [5] and may indicate a technical artifact. We include these models in case it is of interest to simulate reads with these profiles.

4 Coverage Profiles

Figure 3 in the main manuscript illustrates a typical coverage profile for two simulated data sets and the GEUVADIS data set. Here we show a transcript whose coverage profile was unusual in the GEUVADIS data set, so the simulated reads showed coverage profiles

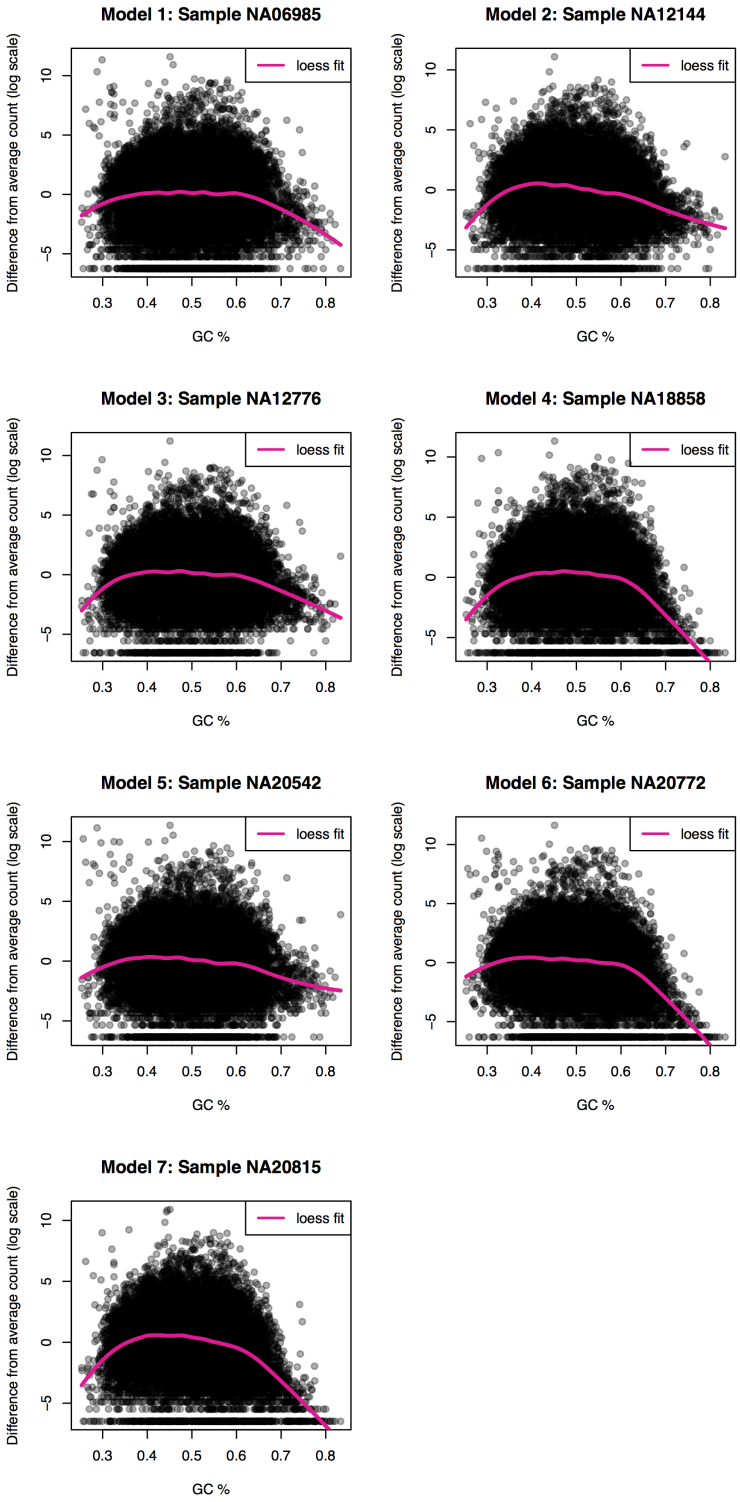

Figure 1: **GC content models for expression included in** *Polyester*. For each of the 7 GEUVADIS replicates (Section 2.2), loess curves were fit to estimate per-transcript deviations from overall mean count based on GC content. In each of these plots, each point represents a transcript, with its GC content percent on the x-axis and its read count on the y-axis. As expected, transcripts with high and low GC content tend to be measured as underexpressed [2, 6, 1]. These models were fit and are illustrated on the $\log_2$ scale: in other words, we added 1 to all transcript counts (to avoid calculating log(0)), log-transformed the counts, then centered the log counts around the mean of all of the log counts. The log-transformation is automatically incorporated in *Polyester* when adding GC bias.

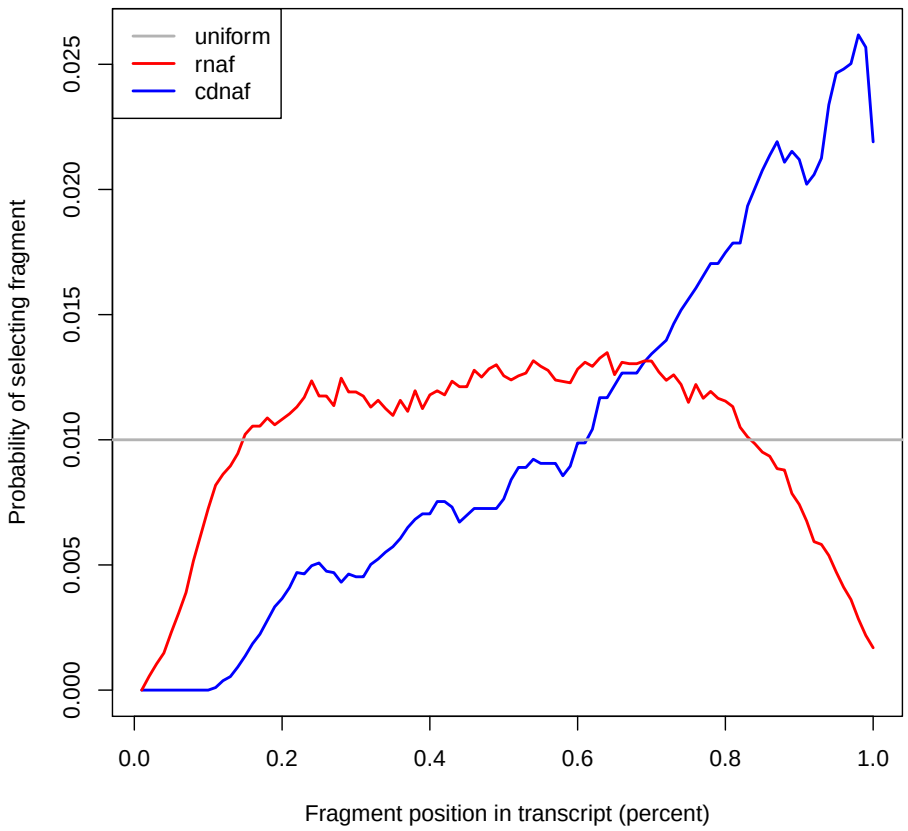

Figure 2: **Positional bias models implemented in** *Polyester*. The figure aims to replicate Supplementary Figure 3 in [4]. Fragment selection across a transcript can be biased based on where the fragment falls in the transcript, as illustrated by the figure. A bias based on RNA fragmentation (`rnaf`) is illustrated in red, while a bias based on cDNA fragmentation (`cdnaf`) is illustrated in blue. The gray line illustrates the uniform, unbiased model, where fragments are equally likely to have originated from any postition in the transcript.

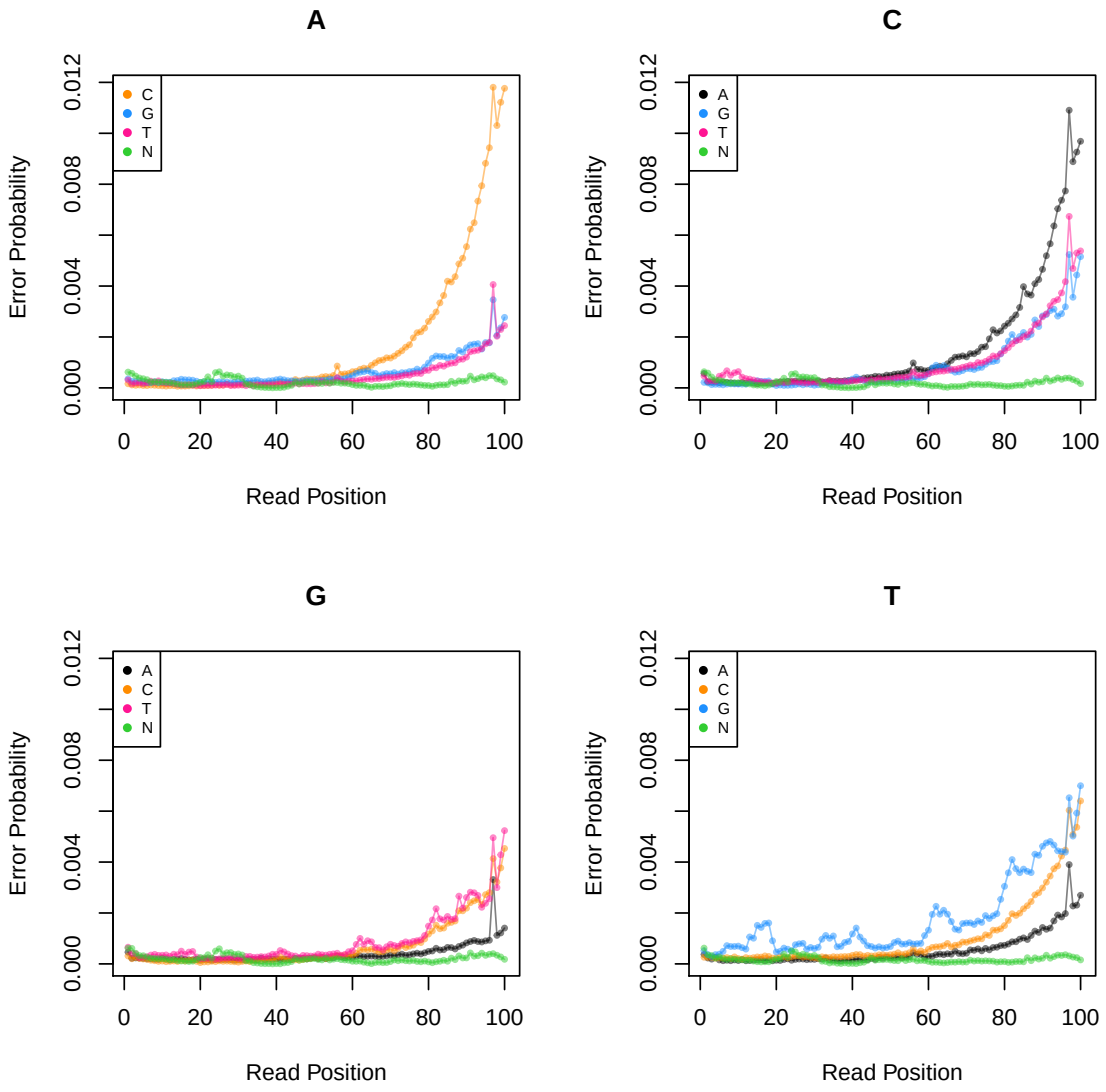

Figure 3: **Empirical error model derived from TruSeq SBS Kit v5-GA chemistry, using Illumina Genome Analyzer IIx, for mate 2 of a paired-end read**.

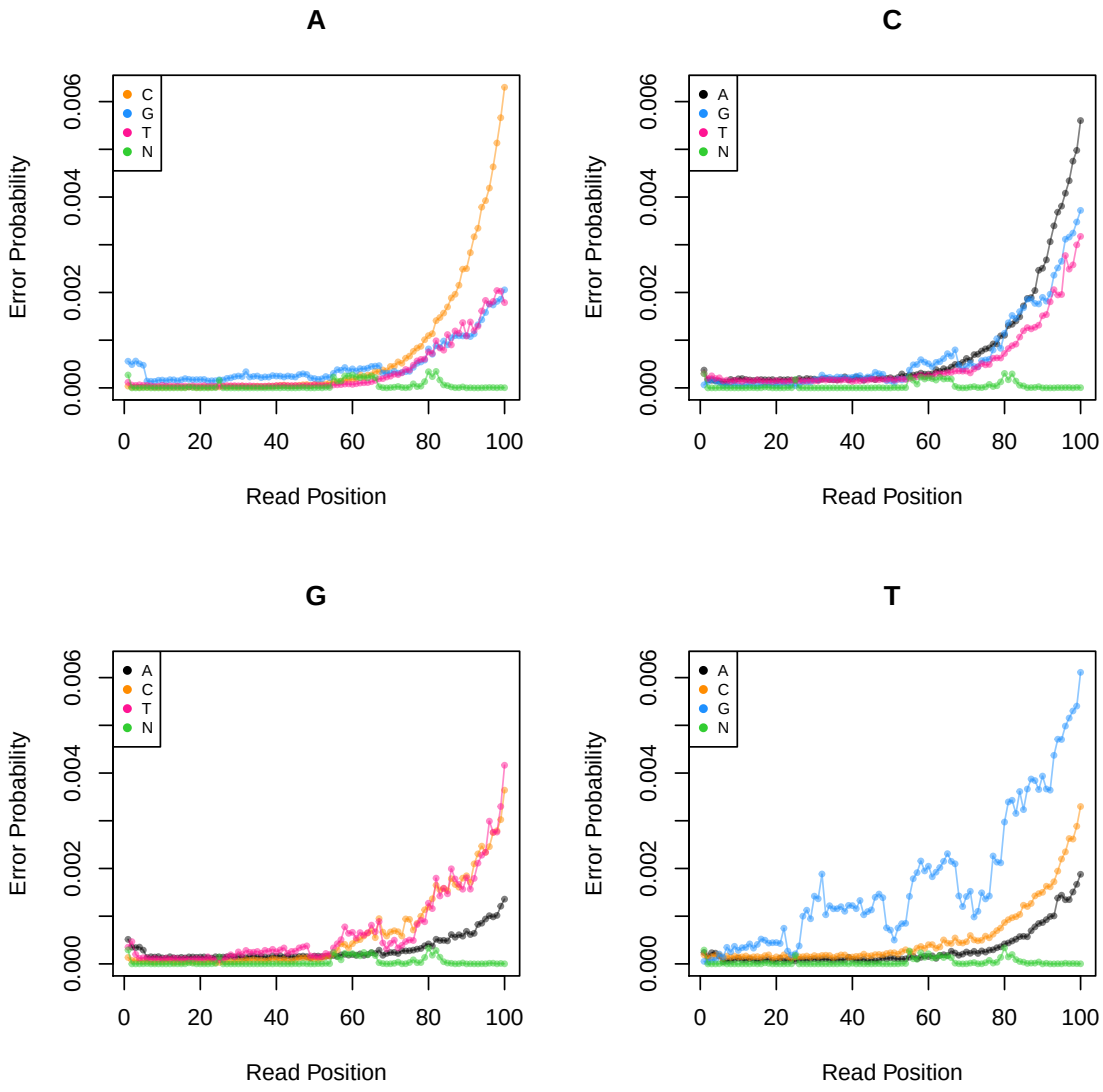

Figure 4: **Empirical error model derived from TruSeq SBS Kit v5-GA chemistry, using Illumina Genome Analyzer IIx, for a single-end read**.

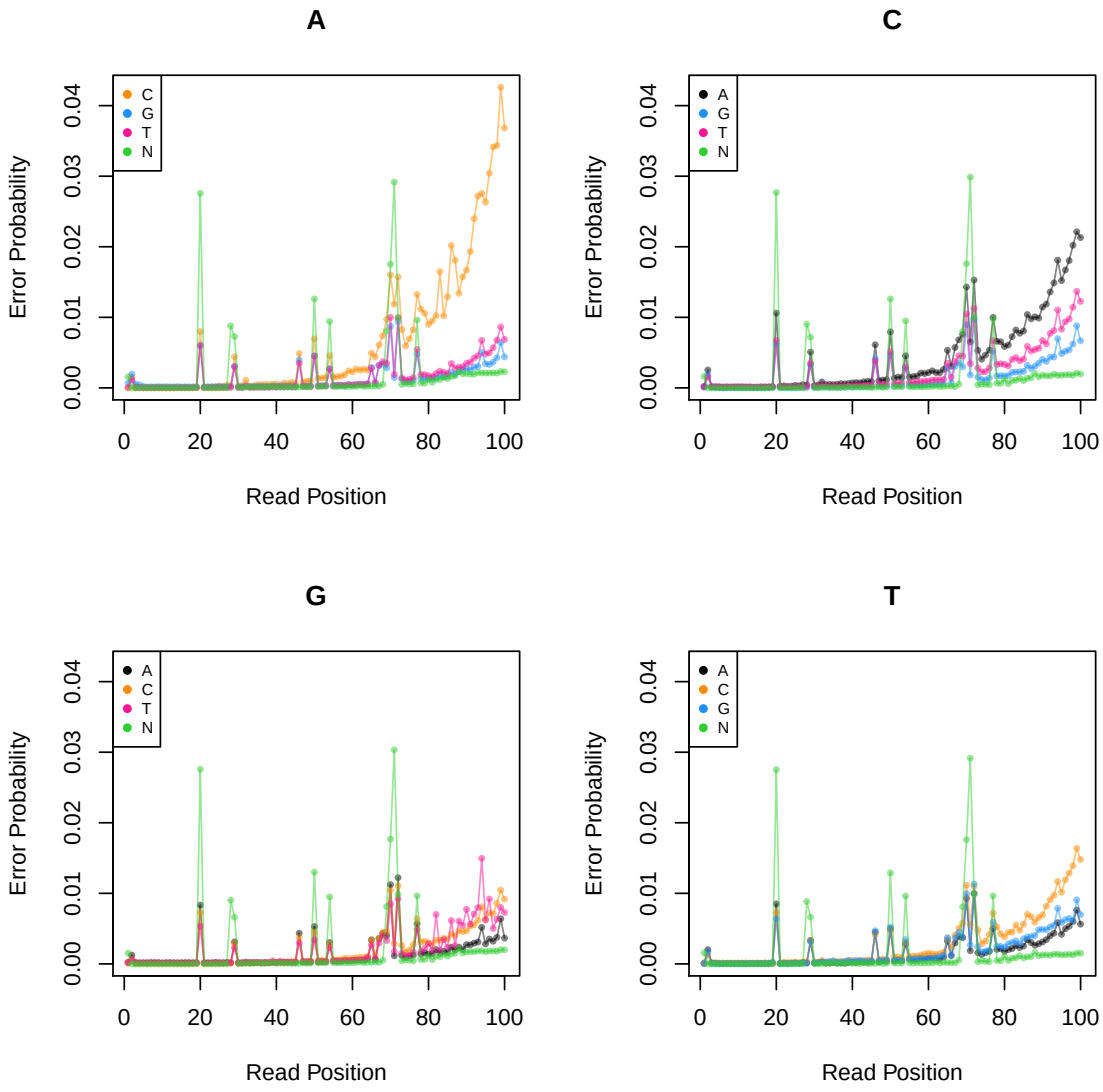

Figure 5: **Empirical error model derived from Illumina Sequencing Kit v4, for mate 1 of a paired-end read**.

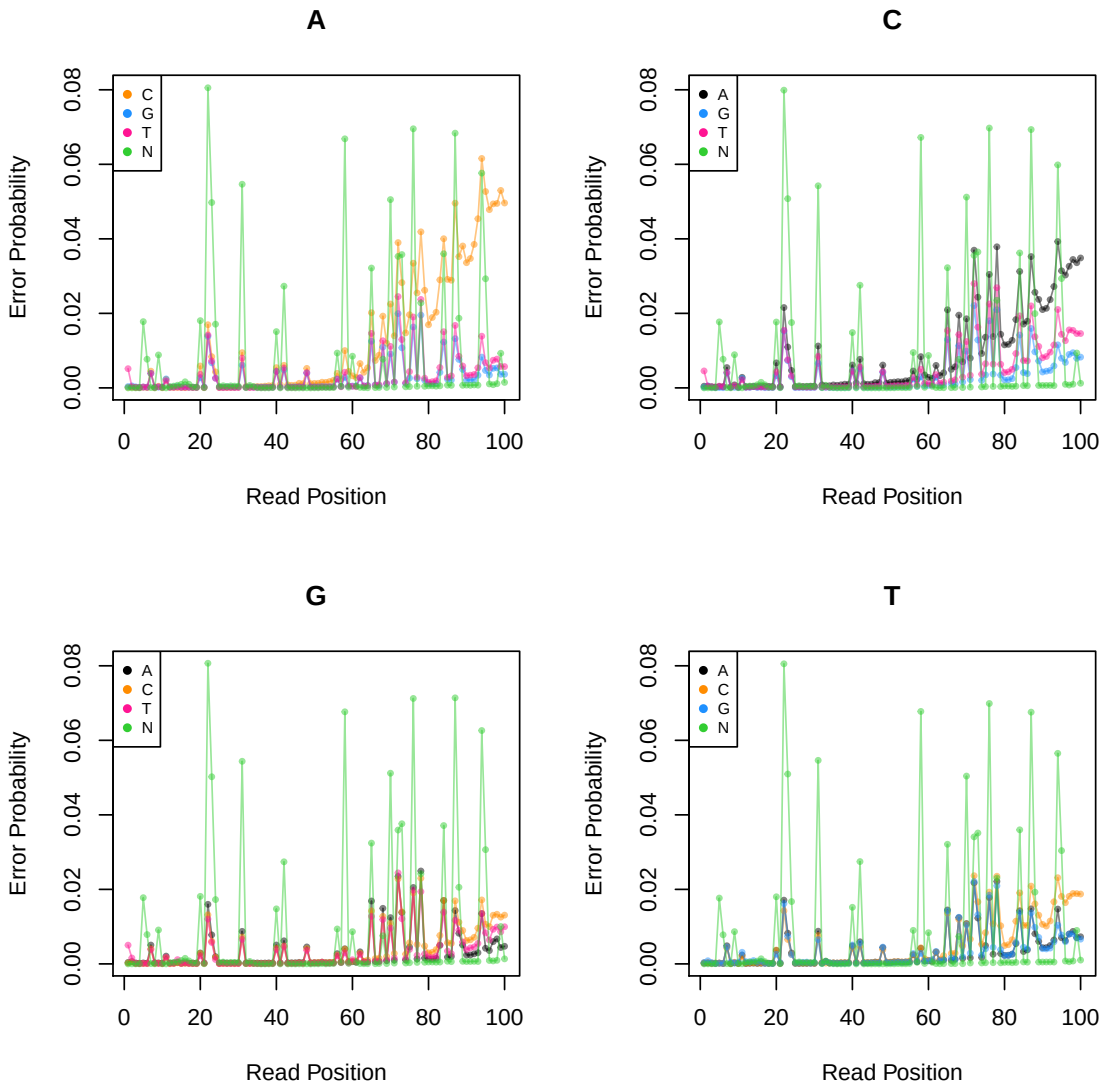

Figure 6: **Empirical error model derived from Illumina Sequencing Kit v4, for mate 2 of a paired-end read**.

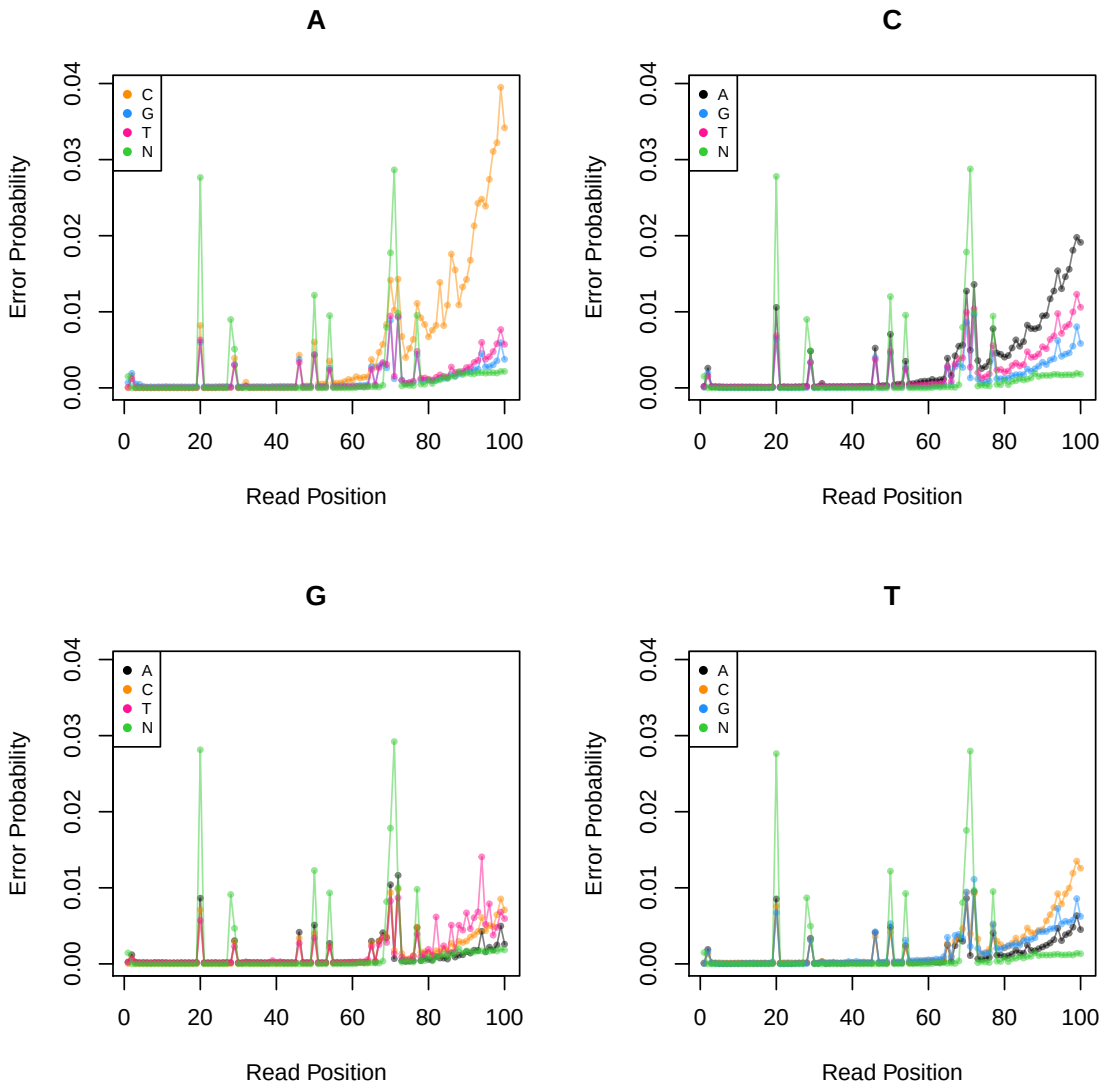

Figure 7: **Empirical error model derived from Illumina Sequencing Kit v4, for a single-end read**.

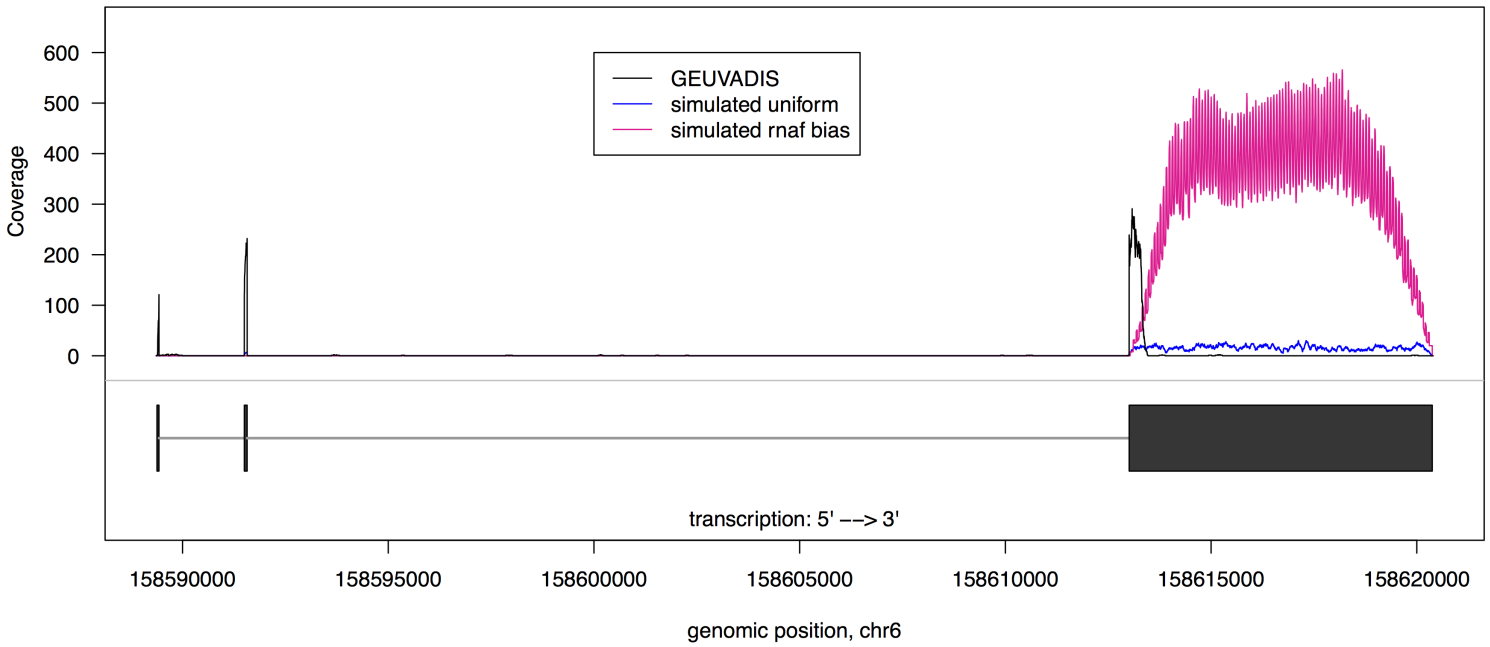

Figure 8: **Unusual simulated coverage profiles.** The bottom panel of this graph illustrates the single isoform of GTF2H5, along with coverage profiles (y-axis) of one replicate from the GEUVADIS data set (black), a data set simulated without positional bias (blue), and a data set simulated with the `rnaf` positional bias model (pink).

that were quite different from the GEUVADIS read coverage profile. This is likely to have occurred because the GEUVADIS data set did not seem to have contained reads from the downstream end of the longest exon of this transcript.

There were 70 coverage plots total generated from this experiment (10 genes, 7 replicates). One is available as Figure 3 in the main text, another is Supplementary Figure 8, and the rest are available for download at `http://figshare.com/articles/Coverage_Plots/1225636`.

5 FPKM correlations

Correlations between transcript-level FPKM estimates estimated for simulated and real data were generally positive. The correlation was very strong when fragments were generated uniformly along the transcript, but weaker when positional bias was added.

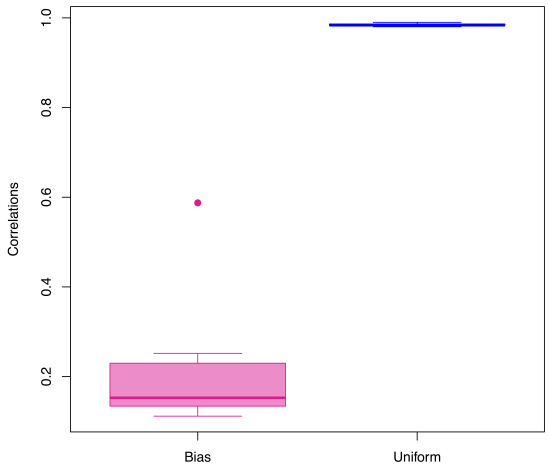

Figure 9: **Correlations between estimated FPKM values in the GEUVADIS data set compared to simulations, with and without postional bias.** Each box in the plot contains 7 correlation measurements, one for each replicate in the study, each of which was obtained by calculating the correlation between FPKM estimates from simulated and GEUVADIS data for the 15 transcripts in the study. Correlations for the simulation with positional bias are shown on the left, and for the simulation with uniform fragmentation (no positional bias) on the right. Correlations were all positive, but were weak in the simulation with bias and very strong in the simulation without bias.

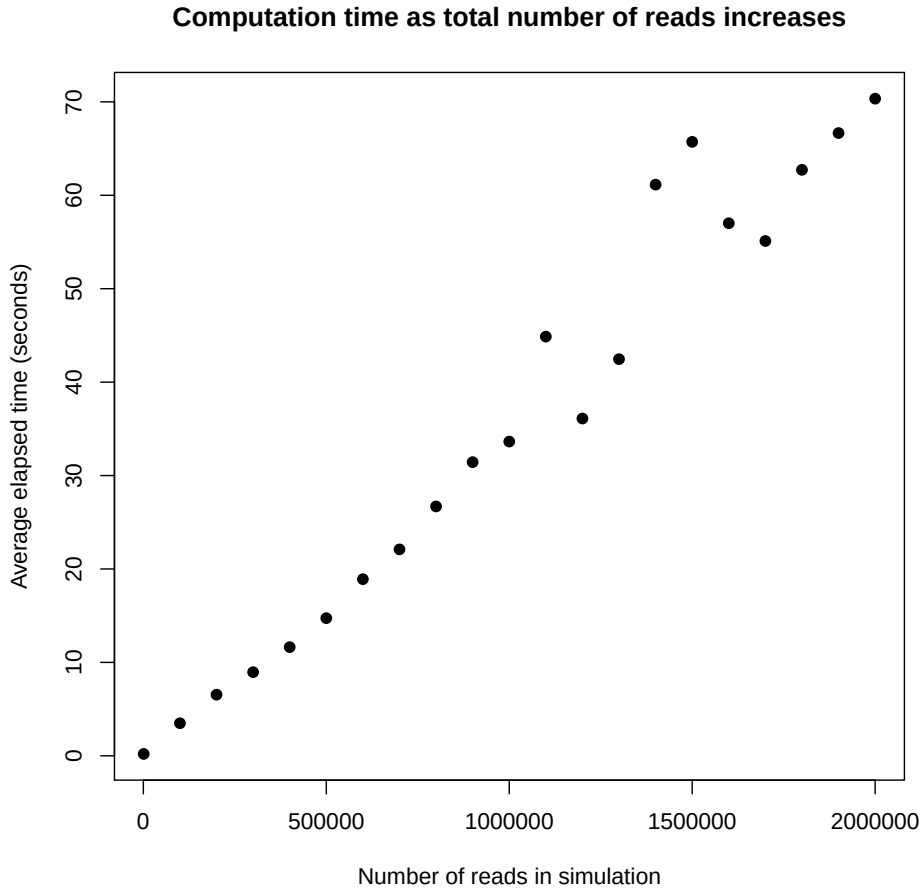

Figure 10: **Linear relationship between total number of reads in experiment and computational time required**. Y-axis represents average elapsed time over 5 replications.

6 Time and Memory Complexity

We ran several benchmarks on Polyester version 1.2.2 to empirically demonstrate the time and memory complexity claims in Section 3.3 of the main manuscript.

The first set of benchmarks shows that Polyester's main simulation function's computation time requirement scales approximately linearly with the total number of reads in the simulation (Supplementary Figure 10). The elapsed time was calculated as an average over 5 calls to the `simulate_experiment` function. One replicate was simulated, and reads were generated approximately equally across 918 transcripts.

The second set of benchmarks shows that `simulate_experiment()` also scales linearly in regards to computational time as the total number of replicates in the experiment increases (11). For this benchmarking analysis, for each replicate, each of 918 transcripts had a baseline mean of 300 reads; the true number of reads simulated from the transcript was drawn from

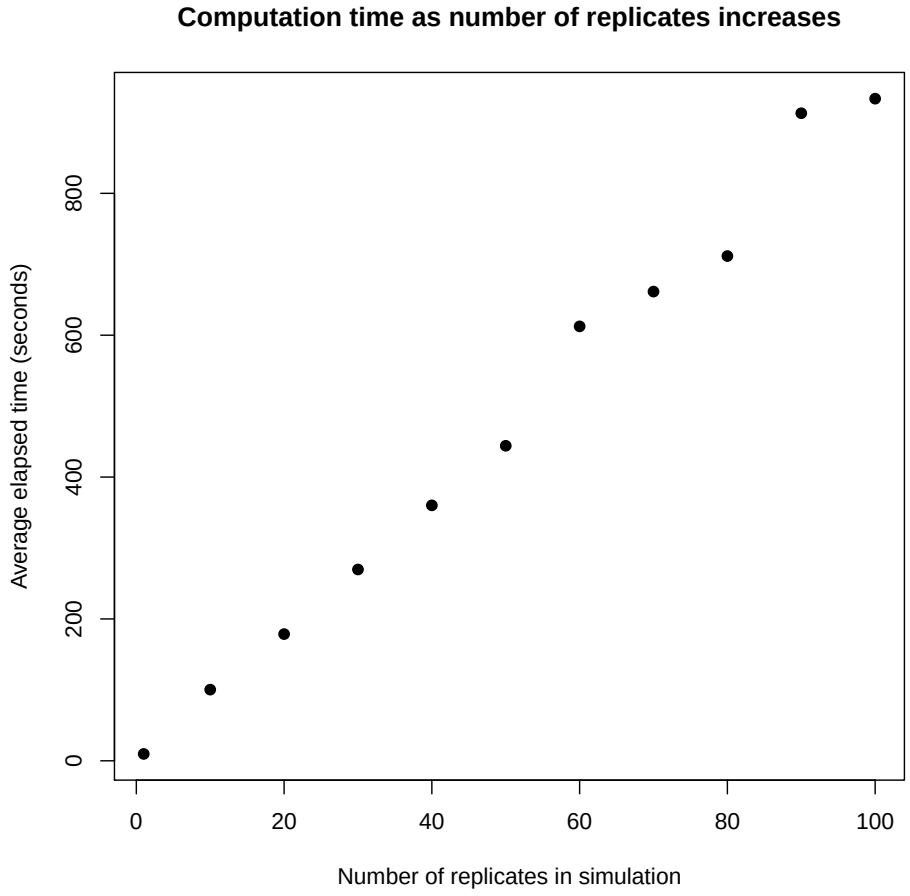

Figure 11: **Linear relationship between number of replicates in experiment and computational time required**. Y-axis represents average elapsed time over 5 replications.

a negative binomial distribution with size parameter 100. No differential expression was simulated for this benchmarking exercise. Like the first set of benchmarks, the elapsed time on the graph was calculated as an average over 5 calls to `simulate_experiment`.

A third set of benchmarks illustrates that simulating positional bias does not add additional computational time burden (Supplementary Figure 12). In the experiment that generated the timings shown in Supplementary Figure 12, we used 918 transcripts and all default settings (baseline mean of 300 reads per transcript, negative binomial size parameter of 100 for all transcripts, uniform error model, etc.) and simulated reads for one replicate. We repeated this five times each for no positional bias, `rnaf` bias, and `cdnaf` bias. There is no noticeable change in elapsed time for either the `rnaf` or `cdnaf` bias models.

Finally, a fourth set of benchmarks shows that using an empirical error model adds a considerable time burden to a simulation (Table 1). Using 918 transcripts and all default parameters (mean of 300 reads per transcript with size=100, no positional bias, etc.) and

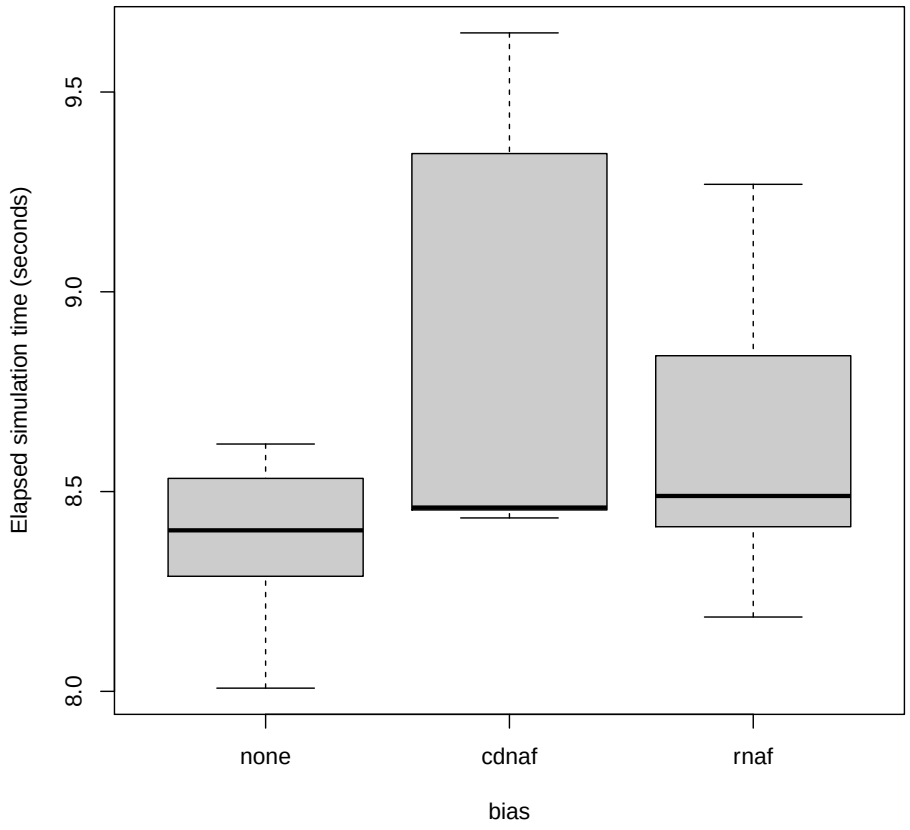

Figure 12: **Effect of positional bias on computational time**. Adding positional bias to the simulation may slow it down very slightly.

Error Model	Rep #	Elapsed Time (sec)
Uniform	1	8.7
	2	8.4
	3	8.6
	4	8.5
	5	8.6
Empirical (Illumina v5)	1	244.1
	2	230.0
	3	246.4
	4	247.5
	5	239.9

Table 1: Computational time estimates comparing uniform error model to empirical Illumina v5 error model.

repeating the benchmark 5 times for both the uniform error model and the empirical Illumina v5 error model, we find that using the `illumina5` error model increases the time required by a factor 27-29. This is not surprising, as the empirical error model is much more detailed and is implemented as an iteration over the bases in the simulation in pure R code rather than a vectorized function. It is a focus for improvement in future versions of the software.

All timing benchmarks were calculated using the `benchmark` function in the *rbenchmark* package in R [3].

An analysis of the maximum amount of memory used by the simulation shows that the amount of memory used scales linearly with the number of reads per replicate in the simulation (Supplementary Figure 13). For this analysis, we ran the same code as we did for the first set of benchmarks, but with a different seed and only doing one replication per scenario (instead of averaging the memory use over five replications). For this figure, we only varied the number of reads simulated for a one-replicate experiment, but replicates are simulated sequentially so we would expect the same pattern if the experiment had multiple replicates and the (maximum) number of reads per replicate were varied.

We also show that memory use is arguably constant in the number of replicates in the experiment (Supplementary Figure 14). These memory estimates were gethered using the same code as in the second set of benchmarks, but again with a different seed and only doing one replication per scenario instead of five. The code for the scenarios in Supplementary Figures 13 and 14 was run on a Linux cluster node in order to take advantage of the Sun Grid Engine scheduling system, which allowed all the scenarios to run in parallel (though there was no paralellization within single scenarios) and automatically recorded the maximum amount of memory used by a single scenario.

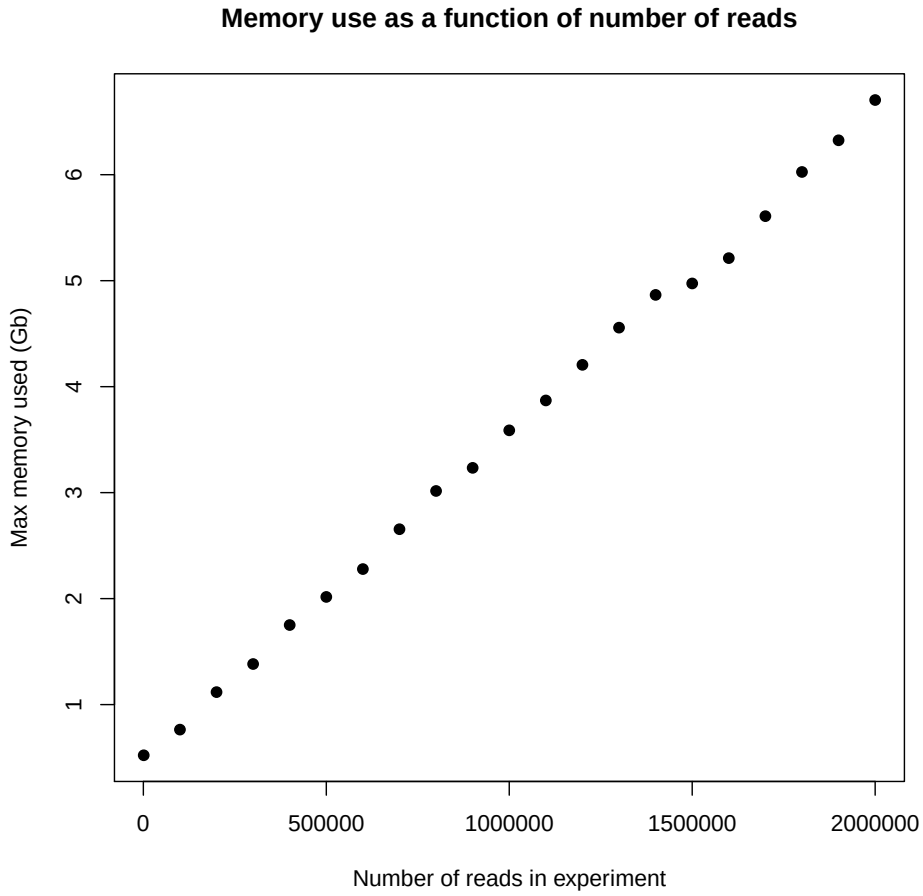

Figure 13: **Linear scaling of max memory usage as number of reads in simulation increases**.

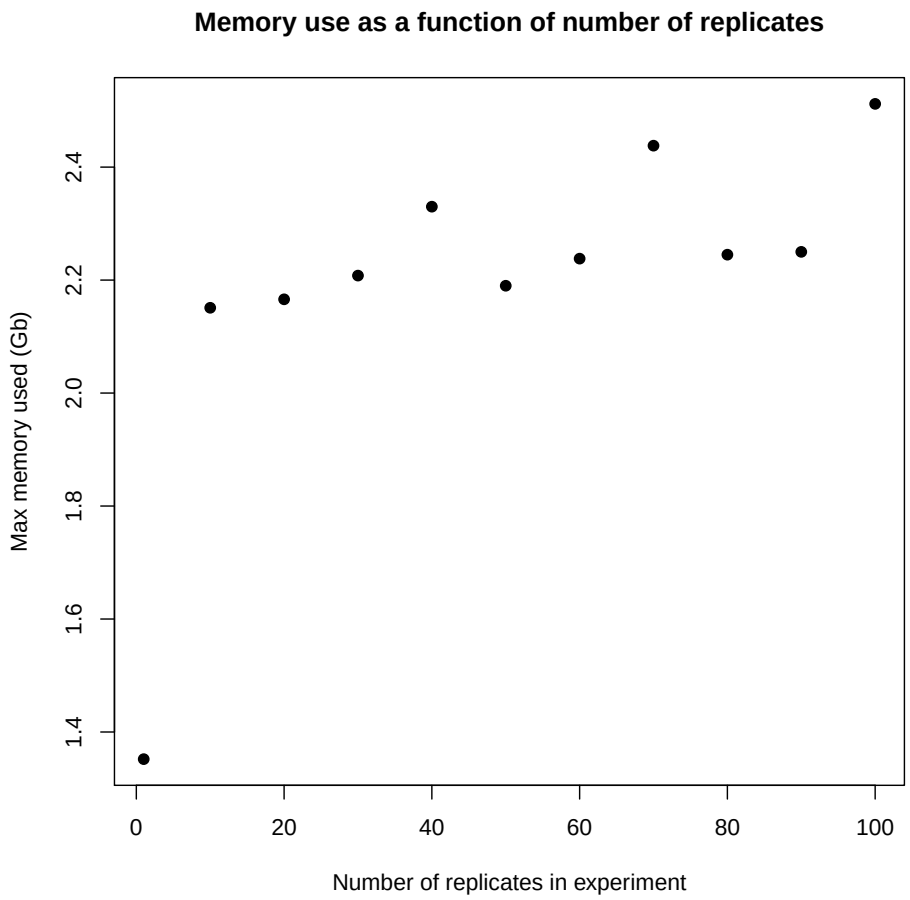

Figure 14: **Max memory usage as number of replicates in simulation increases**. The code is written so that replicates in the simulation are processed sequentially, so Polyester should be $O(1)$ in memory as the number of replicates increases. Here we see the max memory use may increase slightly as the number of replicates increases, but the increase is very small, and we see that doubling the number of replicates in the simulation certainly does not double the memory use.

References

[1] Yuval Benjamini and Terence P Speed. Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Research*, page gks001, 2012.

[2] Kasper D Hansen, Rafael A Irizarry, and WU Zhijin. Removing technical variability in RNA-seq data using conditional quantile normalization. *Biostatistics*, 13(2):204–216, 2012.

[3] Wacek Kusnierczyk. *rbenchmark: Benchmarking routine for R*, 2012. R package version 1.0.0.

[4] Wei Li and Tao Jiang. Transcriptome assembly and isoform expression level estimation from biased RNA-seq reads. *Bioinformatics*, 28(22):2914–2921, 2012.

[5] Kerensa E McElroy, Fabio Luciani, and Torsten Thomas. GemSIM: general, error-model based simulator of next-generation sequencing data. *BMC Genomics*, 13(1):74, 2012.

[6] Davide Risso, Katja Schwartz, Gavin Sherlock, and Sandrine Dudoit. GC-content normalization for RNA-seq data. *BMC Bioinformatics*, 12(1):480, 2011.