Genomics: Generating Ridiculous Networks Since 1978

Camille Scott[1], Eli Zarrindast[2], Luiz Irber[1]

Abstract— **Genomics has pushed the role of computation in biology since its inception, relying heavily on graph theoretic methods to analyze sequencing data. The assembly graph, encompassing both the string graph and de Bruijn graph formulations, has shown great utility in assembling the initially fragmented data and analyzing them both pre- and post- assembly. Previous work has studied sequencing data sets and their resulting assembled contiguous sequences through a statistical framework, allowing estimates for the necessary sequencing to sample an entire genome and the effects of genome features in the physical organism on the resulting assembled sequences; we extend this approach to study the assembly graph itself, starting down a path to fold this statistical framework into the larger graph theoretic approach. We focus on the de Bruijn graph formulation, due to its common use for current sequencing and assembly projects, and take an empirical approach to understanding the graphs generated from known genomes. We present an analysis of the de Bruijn graphs generated by 101 invertebrate genomes, where we measure degree distributions, linear paths, and motif frequency for all subgraphs of size 3 and 4. Our results indicate a decreasing complexity of graphs with respect to their genome size, as well as significant benefits to graph compression for the study of large assembly graphs.**

I. Introduction and Overview

Nucleic acid sequencing technology has had a profound effect on biological research, allowing even small labs to study previously undescribed organisms at the molecular level. Computational approaches have been instrumental in bringing this technology into wide use, due to two primary factors: first, genomes are large and information rich, with the human genome comprising approximately 3 billion base-pairs (3 Gbp); and second, present sequencing instruments are limited to reading several hundred to a few thousand nucleic acid bases at a time, requiring an amplification and random sampling approach to observe and reconstruct complete genomes [1]. Sampling is performed from 1 to 1000-fold redundancy depending on needs, resulting in sequencing experiments comprising up to dozens of terabytes of raw data. The process of resolving these randomly sampled fragments (or "reads") into an approximation of the actual nucleic acid sequence contained in the organism is referred to as assembly.

Many different algorithms have been proposed for solving the assembly problem. Early methods approached it as a string problem, seeking to find a shortest common superstring of the reads; modern approaches have formulated it in a graph theoretic framework [2], chosen for its conceptual simplicity and efficiency in tackling a problem which is otherwise NP-hard to solve optimally. The graphs in this framework are referred to as assembly graphs, and have two formulations: the assembly string graph, which is the graph of overlaps between all sampled reads with transitive edges removed [3], and the de Bruijn graph, which has the set of all subsequences of length k (k-mers) of the reads as its nodes and their length $k-1$ prefix-suffix overlaps as its edges [4]. With both approaches, we extract the sequence corresponding to a set of walks through the graph as our underlying reconstructed sequence; for example, in the de Bruijn graph formulation, we find a (close approximation to) an Eulerian path.

Theoretical approaches to sequencing tracked advances in assembly methods. Early work focused on strategic sequencing, when sequencing was still astronomically expensive and minimizing sequence volume was paramount; this work produced effective models for the number of fragments of a given length and error rate to completely cover (or "close") a genome of a given size [5, 6], and is comparable to early work in the superstring assembly paradigm. As sequencing costs dramatically decreased and graph theoretic approaches were adopted to cope with the data, strategic sequencing became unnecessary, and indeed, modern approaches are often decidedly *un*-strategic, using extreme redundancy to improve results. Research into sequencing theory then mostly aimed to resolve the problem of data size, and data structures for sequencing proliferated [7, 8, 9, 10, 11, 12, 13]. These approaches are characterized by treating the reads as a means to producing assembled sequences; treating the graph itself as a useful object of study is less common. The assembly graph has been studied in order to find good assembly parameters [14], to measure properties of individual genomes [15], and for reducing data redundancy [16], but large-scale analyses of properties across genomes in the vein of the early theoretic work are largely absent. We propose that characterizing assembly graph behavior across a wide range of sequencing data will provide useful insights into assembly graph formulations, assembler heuristics, and assembly-free analyses of biological features.

Section II will go look more closely at the de Bruijn graph assembly formulation and sequencing coverage statistics of Lander, Waterman, and Roach [6, 5]. Section III will discuss our chosen sequencing data sets, our specific de Bruijn graph implementation, our methods for dealing with large graph size, and the graphical properties we measure. We will conclude in Section IV with some of the implications of our results in regards to assembly and assembly graphs, a discussion of the limitations of our approach, and future work we hope to undertake.

[1]Graduate Group for Computer Science, University of California, Davis, Davis, CA camille.scott.w@gmail.com, luiz.irber@gmail.com

[2]Department of Mathematics, University of California Davis, Davis, CA eezarrindast@gmail.com

II. Background

A. Coverage Statistics of Roach, Lander, and Waterman

The coverage statistics for strategic sequencing were fundamental to early understanding of the sequencing process. Early methods for DNA sequencing, such as the chain termination method or Sanger sequencing developed in 1977, were expensive and labor-intensive, while being limited to read of approximately 1000bp; answering the question, "how much do I actually *need* to sequence?" was important for sequencing projects. The resulting coverage statistics remain valid for modern, high-coverage shotgun approaches. Here we will briefly describe them.

Writing F for the read (fragment) length and G for the target genome's length, we may express the probability of covering any given location on the target genome with a fixed read as $\frac{F}{G}$. This formulation assumes $F \ll G$, which is often but not always the case. The probability that a fixed location on the genome was not covered by the read is then

$$1 - \frac{F}{G}$$

for one fixed read and

$$\left[1 - \frac{F}{T}\right]^N$$

for N fixed reads. The probability of at least one read covering a given location on the target may therefore be written

$$P = 1 - \left[1 - \frac{F}{G}\right]^N.$$

In most real-world cases, $N \gg 1$. So we may write:

$$P = \left[1 - \frac{L}{G}\right]^N \sim e^{-N\frac{F}{G}}$$

where $R = N\frac{F}{G}$ is called the *redundancy*. The redundancy represents the average number of times a position is (totally) covered by reads. P is the same as the expected value of the random variable C, which represents the fraction of the target coverage. One will often see the final result quoted in use as:

$$E\langle C\rangle = 1 - e^{-R}$$

This expression predicts that *coverage*, in arbitrary case, progresses along a curve that is a function of number of reads, read length, and fragment size.

These statistics remain important for our approach for two reasons. Firstly, they describe the relationship between the reads and the underlying genome they sample, which is fundamental in understanding the relationship between the assembly graph and the underlying set of paths representing the true genome. Secondly, we can convert the coverage statistics for reads of length F directly to a model for de Bruijn graph coverage, where the k-mer length is equal to F and nodes in the graph have a count associated with them. We will explore de Bruijn graphs next.

B. de Bruijn Graph Formulation

The de Bruijn graph is one of the two primary formulations for the assembly graph. The de Bruijn graph for sequence assembly is a more constrained case of the generalized version, which is as follows: given some alphabet Σ of size m, the de Bruin graph of dimension n has m^n nodes, one for each possible length-n sequence of the symbols in Σ. An edge is drawn between two nodes u and v if the length $n-1$ symbol suffix of u is the same as the length $n-1$ prefix of v [17].

For our formulation, we will use $\Sigma = \{A, T, C, G\}$. Traditionally, we use k instead of n for our dimensionality, and the sequence corresponding to each node is called a k-mer. The complete de Bruijn graph over length k has $\frac{4^k}{2}$ possible nodes, the division by 2 arising from the reverse-complement nature of DNA.[1] In the language of bond percolation, a node is active if its k-mer exists in one or more reads in the input sample, and the set of edges is that induced by the active nodes' prefix-suffix overlaps as described above. The important feature to consider is that this computes overlaps between reads and stores edges implicitly: any assembly de Bruijn graph can be represented by a simple set data structure and can be traversed given a starting k-mer. In other formulations, we would be forced make N^2 string comparisons in the worst case to compute the overlap graph, which is computationally prohibitive for genomes of reasonable size.

Once we have decomposed all the reads into k-mers and inserted them into the graph, we can move forward with extracting the assembled sequence or performed graphical analysis. Assembly is accomplished in theoretical terms by finding an Eulerian path through the graph and outputting the concatenated unique bases at each node; in reality, and Eulerian path can rarely be found, and heuristic methods must be employed [12]. This is due firstly to sequencing error, where we can expect a single nucleotide to be misread with probability $0.005 - 0.01$ for commonly used shotgun instruments, and secondly to repetitive and low-complexity regions. Many genes are duplicated within a genome, and regulatory regions often have low information content which leads, perhaps counter-intuitively, to more complex de Bruijn graphs [18]. The major implication of the de Bruijn graph formulation is that it collapses repetitions into their non-redundant nodes and a cycle, a consequence of their indexing and storage efficiency. Many methods have been developed to overcome these limitations and improve assembly quality which readers can explore further in Jared Simpson's excellent review paper [1].

We are particularly interested in de Bruijn graphs outside of their utility for assembly. de Bruin graph-based, and more generally k-mer based, analyses have proven useful for a number of problems surrounding genomics. Digital normalization is a k-mer based technique for reducing the

[1] In DNA, each nucleotide has a complementary base $\{A \to T, C \to G\}$. The reverse complement of a sequence is obtained by performing the previous transformation and reversing the sequence. This phenomenon is the byproduct of the double-helix structure: the primary "strand" is one side of the latter, the reverse complement the other.

size of a sequencing sample use the de Bruijn graph: by tracking the counts of each k-mer, a read can be discarded if the median of its k-mer counts is above a certain threshold, a fast proxy for determining the coverage of its corresponding genomic region [16]. Latent Strain Analysis used the de Bruijn graph along with a locality-sensitive hashing scheme to find microbial genomes and communities in metagenomic[2] data without assembling a reference [19], building on a foundation of other metagenomics work using k-mers for classification [20].

III. Technical Approach and Results

A. Collecting data

We downloaded all invertebrate and mammalian complete genomes available on NCBI. This comprised approximately 84 GB total data, with 10 GB (101 genomes) from invertebrates and about 74 GB (99 genomes) mammalian. We started with the mammalian genomes, which are usually bigger (and more complex) than invertebrates. Unfortunately this data was hard to use as test case for methods, since each iteration consumed too many resources, or didn't even complete in our machines (with 8-32 CPUs and 60GB of RAM). Invertebrate genomes, on the other hand, are a tractable investigation point, since most genomes are smaller and the largest one (Octopus) is in range of smaller mammalian genomes. We want to analyze the de Bruijn graphs of these genomes in order to compare their properties. but even using the invertebrate genomes we still wish to significantly reduce the size of the data that we are working with, so we compressed graphs before analyzing them.

B. Generating de Bruijn Graphs

Finding out what k-mer size will be appropriate for a given genome is often a heuristic process, and in this case we found that using smaller k-mers was taking too long on our computers [14]. We decided to use 31-mers, but the approach is easily extensible to any other k-mer value. We created the de Bruijn graph for each genome by hashing each 31-mer from the assembled sequences using a Bloom filter, a space-efficient probabilistic data structure created by Burton Howard Bloom in 1970. Bloom filters test whether an element is a member of a set, with the caveat that false positive matches are possible, but false negatives are not. Querying an element effectively returns either "probably in set" or "not in set". In general, a hash function has input any length, and output an alphanumeric string of a usually lesser length which is supposed to uniquely identify the associated input. For an elementary introduction to Bloom filters and hashes, see [21].

Nevertheless, the graphs we attempted to assemble from the collection of 31-mers were very big. The smallest, with 480K nodes, was the Russian wheat aphid. The largest was the octopus, with 1.7B nodes. On average, the graphs had 280 M nodes, which is prohibitive for most analysis using traditional network theory libraries. See figure 2 for the statistics on these graphs.

[2]A metagenomic project sequences all DNA in an environment, such as a digestive tract, soil, or water, in an effort to characterize the genomes of all microbes present. These data are significantly larger than single genome data sets, generating assembly graphs with potentially hundreds of billions of nodes.

C. Compressing the Graph Data

One way to reduce the number of nodes in de Bruijn graphs is collapsing their linear paths. A linear path is composed by degree-2 nodes, and they can be merged into a single degree-2 node and still maintain the de Bruijn graph properties, but allowing nodes to represent multiple k sizes. To do this, we use the de Bruijn graph encoded in the Bloom filter and generate a explicit representation using Graph Modeling Language (GML), a hierarchical ASCII-based file format for describing graphs. [22] This representation also allowed us to use conventional traditional network analysis tools and programs should we desire.

The compression method greatly reduces the number of nodes in our graphs. This time the smallest was 40K (Western honey bee), the largest 98M (Atlantic horseshoe crab), and the average was 11M nodes. Overall, the average reduction in size was 33-fold, with a maximum of 434-fold.

D. Graph Properties

The first things we checked about our de Bruijn graphs were the conventional graph statistics. We looked at the average node degree, average number of nodes, and average number of edges over all nodes/edges in (the elements of) the set of graphs. See figures 1 and 2 .

Next, we did motif analysis on the graphs. A motif is a configuration of connection between a small set of nodes, that may be found perhaps many times in a larger graph. The motifs that we considered are the isomorphism classes for connected subgraphs with 3 and 4 nodes, as looking at 5-node motifs turned out to be too large a computational load. Using *graph_tool.clustering.motifs* to explore, we searched for the default motifs first (figure 3). The library does however support user-provided motifs.

When present, the 3- and 4-node motifs correspond to repetitive elements in the genome. There are organisms whose genomes are actually highly repetitive, rather than it being a sampling distortion; one example is corn (Zea mays), whose genome is 80 percent repetitive [23]. Motifs appear in graph-theoretic analysis as representations of graph complexity, like smaller cycles or branches. We analyzed the correlations between motif isoform classes (figure 5).

IV. Discussion and Conclusions

We examined the typical vertex and node characteristics across a set of 101 graphs of genomes of invertebrates. We compressed this data to a workable size, then examined the graphs for graph motifs as well as searching for and counting linear paths. The approach is extensible to other assembled genomes and also to raw reads.

It is difficult to know what the best k is when making a graph from k-mers. For this reason there may be genomes

whose graphs could easily be analyzed with different k values, or there may be certain properties in graphs we analyzed that are "hiding" because we never got to them. During the method development we focused on a small number of genomes, but now we can extend the analysis using clusters instead of our servers with limited resources.

Based on knowledge of biological features of the organisms, we also like to make our own motifs to search for; these motifs can be more representative or telling than the ones we chose for this analysis, which just used the default isomorphism classes for graphs with connected 3 and 4 nodes. And finally, at all times one should keep in mind the coverage statistics that we reviewed earlier: we are not getting perfect or 1-to-1 coverage of these genomes when we read from them, and it would be easy for a procedural error or bias to slip into the process undetected on the assumption that it is an actual feature of the target genome.

We would like to repeat the analysis to address these limitations, but using sequencing reads instead of assembled genomes, and also varying the k-mer size, We can compare results from assembled genomes and sequencing reads and see how motif abundance changes, and seeing what generated more or fewer linear paths, and based on these metrics also discover any technical errors in assembly practices.

Another possibility is setting the problem as a multilayer (or multiplex) network, since we can map the nodes from one genome directly to the same nodes in other genomes (each genome being a layer, in multiplex terms), with a possible distance metric between layers being the phylogenetic distance between species.

Finally, we are working on a method to find common properties in genome assembly graphs, for gaining a better understanding on how to model genomes and then use these models to make decisions during graph construction. Successful outcomes would also improve existing genome assembly methods, as well as establishing techniques in assembly-free comparative genomics. There would be considerable insight into the assessment of repetitive structures: in particular transposons, retrotransposons, LTRs, etc. Assembly-free methods are especially useful in the context of increasing data generation and cost reduction in sequencing, and allow analysis which are simply not viable with current methods.

REFERENCES

[1] Jared T. Simpson and Mihai Pop. The Theory and Practice of Genome Sequence Assembly. *Annual Review of Genomics and Human Genetics*, 16(1):153–172, 2015. doi: 10.1146/annurev-genom-090314-050032. URL http://dx.doi.org/10.1146/annurev-genom-090314-050032.

[2] Eugene W. Myers. Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, 2(2):275–290, 1995. URL http://online.liebertpub.com/doi/abs/10.1089/cmb.1995.2.275.

[3] E. W. Myers. The fragment assembly string graph. *Bioinformatics*, 21(Suppl 2):ii79–ii85, September 2005. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/bti1114. URL http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/bti1114.

[4] P. A. Pevzner, H. Tang, and M. S. Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, August 2001. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.171285098. URL http://www.pnas.org/cgi/doi/10.1073/pnas.171285098.

[5] J. C. Roach. Random subcloning. *Genome Research*, 5(5):464–473, December 1995. ISSN 1088-9051, 1549-5469. doi: 10.1101/gr.5.5.464. URL http://genome.cshlp.org/content/5/5/464.

[6] E. S. Lander and M. S. Waterman. Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*, 2(3):231–239, April 1988. ISSN 0888-7543.

[7] Jason Pell, Arend Hintze, Rosangela Canino-Koning, Adina Howe, James M. Tiedje, and C. Titus Brown. Scaling metagenome sequence assembly with probabilistic de Bruijn graphs. *Proceedings of the National Academy of Sciences*, 109 (33):13272–13277, 2012. URL http://www.pnas.org/content/109/33/13272.short.

[8] H. Li and R. Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, July 2009. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btp324. URL http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btp324.

[9] Phillip E. C. Compeau, Pavel A. Pevzner, and Glenn Tesler. How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, 29(11):987–991, November 2011. ISSN 1087-0156. doi: 10.1038/nbt.2023. URL http://www.nature.com/nbt/journal/v29/n11/abs/nbt.2023.html.

[10] J. T. Simpson and R. Durbin. Efficient construction of an assembly string graph using the FM-index. *Bioinformatics*, 26(12):i367–i373, June 2010. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btq217. URL http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btq217.

[11] Z. Li, Y. Chen, D. Mu, J. Yuan, Y. Shi, H. Zhang, J. Gan, N. Li, X. Hu, B. Liu, B. Yang, and W. Fan. Comparison of the two major classes of assembly algorithms: overlap-layout-consensus and de-bruijn-graph. *Briefings in Functional Genomics*, 11(1):25–37, January 2012. ISSN 2041-2649, 2041-2657. doi: 10.1093/bfgp/elr035. URL http://bfgp.oxfordjournals.org/cgi/doi/10.1093/bfgp/elr035.

[12] D. R. Zerbino and E. Birney. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821–829, February 2008. ISSN 1088-9051. doi: 10.1101/gr.074492.107. URL http://www.genome.org/cgi/doi/10.1101/gr.074492.107.

[13] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J.M. Jones, and I. Birol. ABySS: A parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, June 2009. ISSN 1088-9051. doi: 10.1101/gr.089532.108. URL http://genome.cshlp.org/cgi/doi/10.1101/gr.089532.108.

[14] Rayan Chikhi and Paul Medvedev. Informed and automated k-mer size selection for genome assembly. *Bioinformatics*, 30(1):31–37, January 2014. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btt310. URL http://bioinformatics.oxfordjournals.org/content/30/1/31.

[15] Jared T. Simpson. Exploring genome characteristics and sequence quality without a reference. *Bioinformatics*, 30(9):1228–1235, May 2014. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btu023. URL http://bioinformatics.oxfordjournals.org/content/30/9/1228.

[16] C Titus Brown, Adina Howe, Qingpeng Zhang, Alexis B Pyrkosz, and Timothy H Brom. A reference-free algorithm

for computational normalization of shotgun sequencing data. *arXiv preprint arXiv:1203.4802*, 2012.

[17] Bruijn, de NG Dick. A combinatorial problem. *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam*, 49(7):758, 1946. URL http://repository.tue.nl/597473.

[18] Guy Bresler, Ma'ayan Bresler, and David Tse. Optimal assembly for high throughput shotgun sequencing. *BMC Bioinformatics*, 14(5):1–13, 2013. ISSN 1471-2105. doi: 10.1186/1471-2105-14-S5-S18. URL http://dx.doi.org/10.1186/1471-2105-14-S5-S18.

[19] Detection of low-abundance bacterial strains in metagenomic datasets by eigengenome partitioning : Nature Biotechnology : Nature Publishing Group. URL http://www.nature.com/nbt/journal/v33/n10/abs/nbt.3329.html.

[20] Derrick E. Wood and Steven L. Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol*, 15(3):R46, 2014. URL http://www.biomedcentral.com/content/pdf/gb-2014-15-3-r46.pdf.

[21] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet mathematics*, 1(4): 485–509, 2004. URL http://www.tandfonline.com/doi/abs/10.1080/15427951.2004.10129096.

[22] Michael Himsolt. GML: A portable graph file format. *Html page under http://www. fmi. uni-passau. de/graphlet/gml/gml-tr. html, Universitt Passau*, 1997. URL http://www.uni-passau.de/fileadmin/files/lehrstuhl/brandenburg/projekte/gml/gml-technical-report.pdf.

[23] Patrick S. Schnable, Doreen Ware, Robert S. Fulton, Joshua C. Stein, Fusheng Wei, Shiran Pasternak, Chengzhi Liang, Jianwei Zhang, Lucinda Fulton, Tina A. Graves, Patrick Minx, Amy Denise Reily, Laura Courtney, Scott S. Kruchowski, Chad Tomlinson, Cindy Strong, Kim Delehaunty, Catrina Fronick, Bill Courtney, Susan M. Rock, Eddie Belter, Feiyu Du, Kyung Kim, Rachel M. Abbott, Marc Cotton, Andy Levy, Pamela Marchetto, Kerri Ochoa, Stephanie M. Jackson, Barbara Gillam, Weizu Chen, Le Yan, Jamey Higginbotham, Marco Cardenas, Jason Waligorski, Elizabeth Applebaum, Lindsey Phelps, Jason Falcone, Krishna Kanchi, Thynn Thane, Adam Scimone, Nay Thane, Jessica Henke, Tom Wang, Jessica Ruppert, Neha Shah, Kelsi Rotter, Jennifer Hodges, Elizabeth Ingenthron, Matt Cordes, Sara Kohlberg, Jennifer Sgro, Brandon Delgado, Kelly Mead, Asif Chinwalla, Shawn Leonard, Kevin Crouse, Kristi Collura, Dave Kudrna, Jennifer Currie, Ruifeng He, Angelina Angelova, Shanmugam Rajasekar, Teri Mueller, Rene Lomeli, Gabriel Scara, Ara Ko, Krista Delaney, Marina Wissotski, Georgina Lopez, David Campos, Michele Braidotti, Elizabeth Ashley, Wolfgang Golser, HyeRan Kim, Seunghee Lee, Jinke Lin, Zeljko Dujmic, Woojin Kim, Jayson Talag, Andrea Zuccolo, Chuanzhu Fan, Aswathy Sebastian, Melissa Kramer, Lori Spiegel, Lidia Nascimento, Theresa Zutavern, Beth Miller, Claude Ambroise, Stephanie Muller, Will Spooner, Apurva Narechania, Liya Ren, Sharon Wei, Sunita Kumari, Ben Faga, Michael J. Levy, Linda McMahan, Peter Van Buren, Matthew W. Vaughn, Kai Ying, Cheng-Ting Yeh, Scott J. Emrich, Yi Jia, Ananth Kalyanaraman, An-Ping Hsia, W. Brad Barbazuk, Regina S. Baucom, Thomas P. Brutnell, Nicholas C. Carpita, Cristian Chaparro, Jer-Ming Chia, Jean-Marc Deragon, James C. Estill, Yan Fu, Jeffrey A. Jeddeloh, Yujun Han, Hyeran Lee, Pinghua Li, Damon R. Lisch, Sanzhen Liu, Zhijie Liu, Dawn Holligan Nagel, Maureen C. McCann, Phillip SanMiguel, Alan M. Myers, Dan Nettleton, John Nguyen, Bryan W. Penning, Lalit Ponnala, Kevin L. Schneider, David C. Schwartz, Anupma Sharma, Carol Soderlund, Nathan M. Springer, Qi Sun, Hao Wang, Michael Waterman, Richard Westerman, Thomas K. Wolfgruber, Lixing Yang, Yeisoo Yu, Lifang Zhang, Shiguo Zhou, Qihui Zhu, Jeffrey L. Bennetzen, R. Kelly Dawe, Jiming Jiang, Ning Jiang, Gernot G. Presting, Susan R. Wessler, Srinivas Aluru, Robert A. Martienssen, Sandra W. Clifton, W. Richard McCombie, Rod A. Wing, and Richard K. Wilson. The B73 Maize Genome: Complexity, Diversity, and Dynamics. *Science*, 326(5956):1112–1115, November 2009. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1178534. URL http://science.sciencemag.org/content/326/5956/1112.

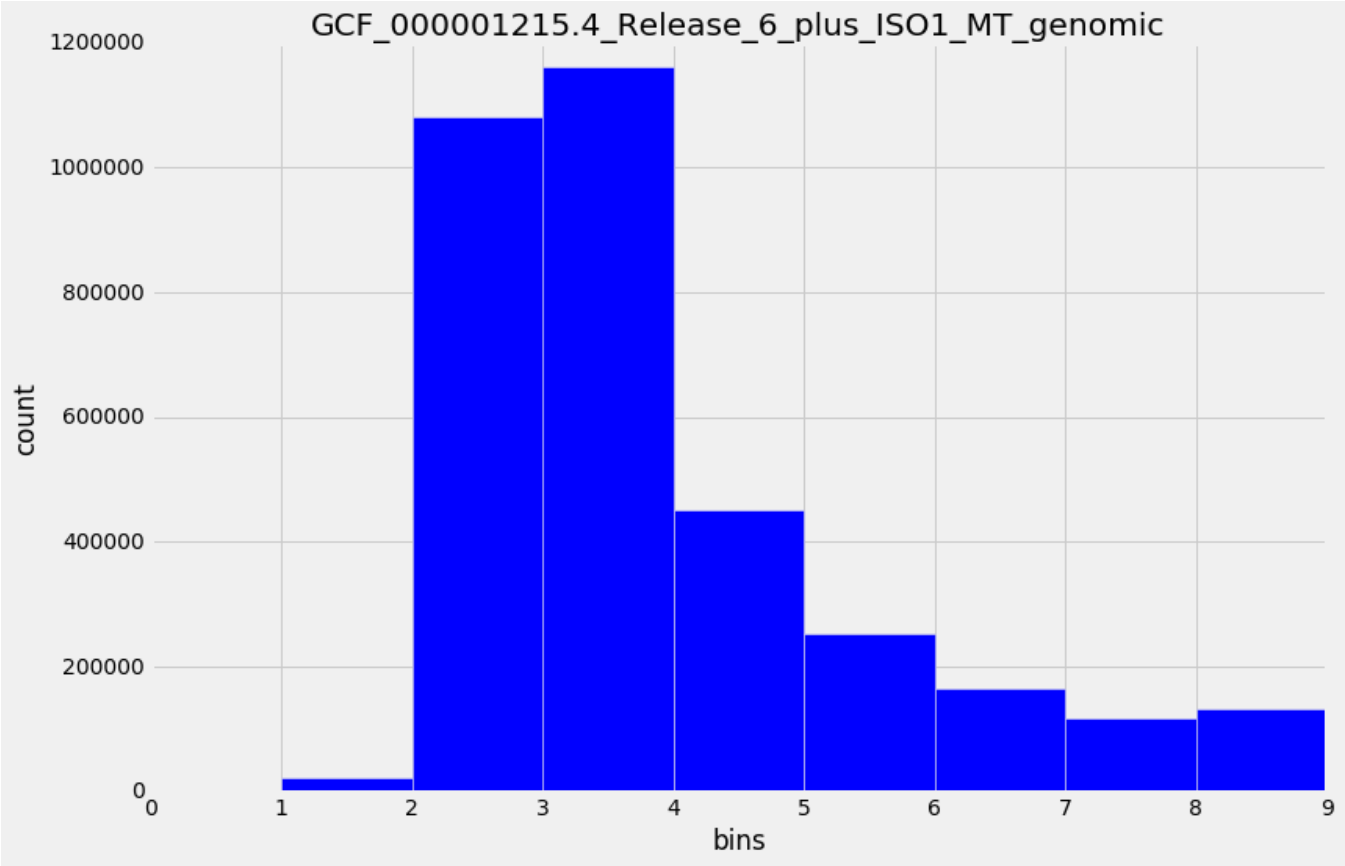
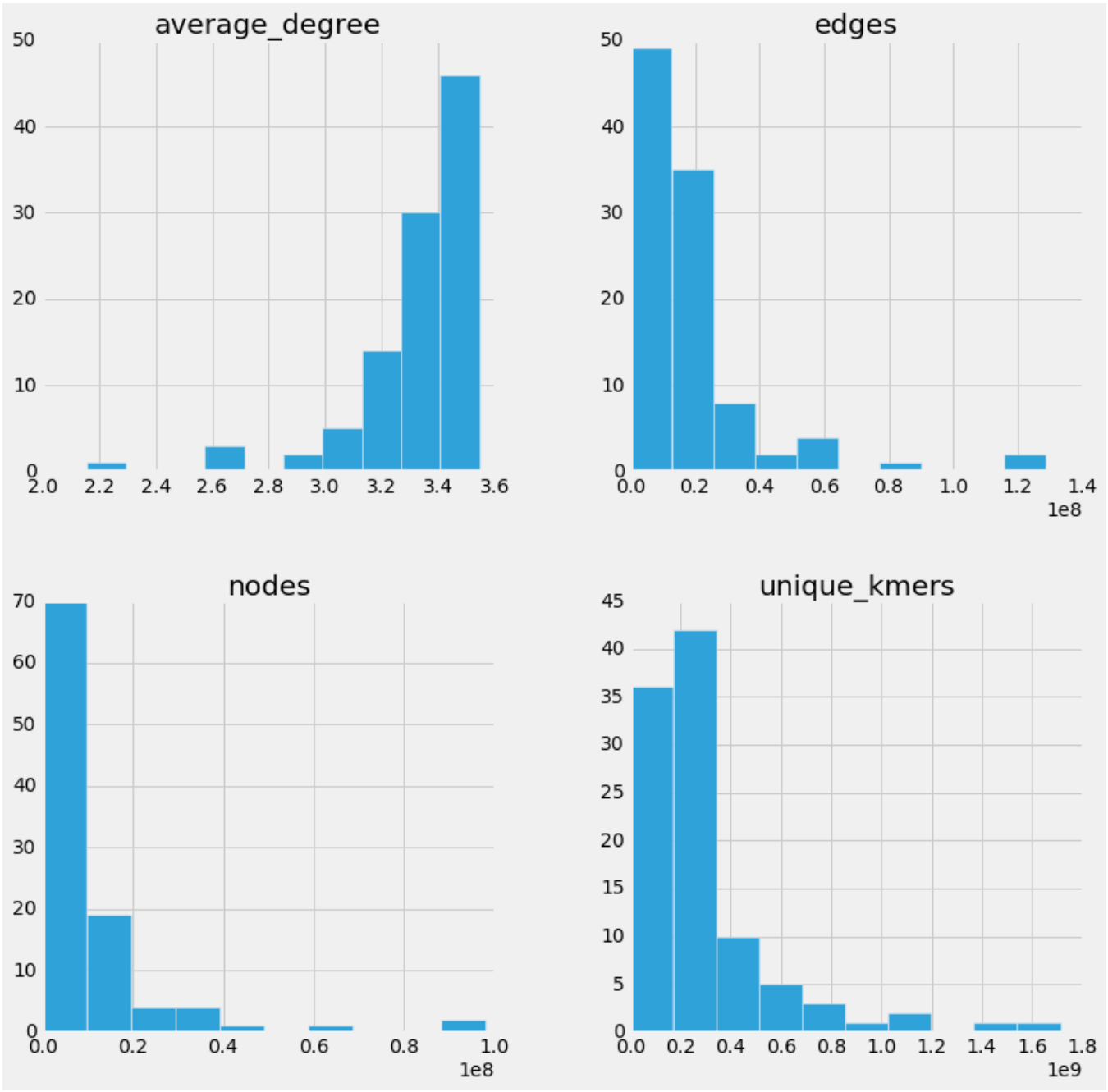


Fig. 1. Degree distribution for a specific organism, Drosophila melanogaster (fruit fly). Most nodes have degree 2 and 3.



.

Fig. 2. Histograms for all 101 invertebrate genomes. Nodes are the vertices in the compressed de Bruijn graph, unique k-mers are the vertices in the uncompressed dBg. Note the reduction in the number of nodes in the compressed representation, two orders of magnitude smaller than the uncompressed one.

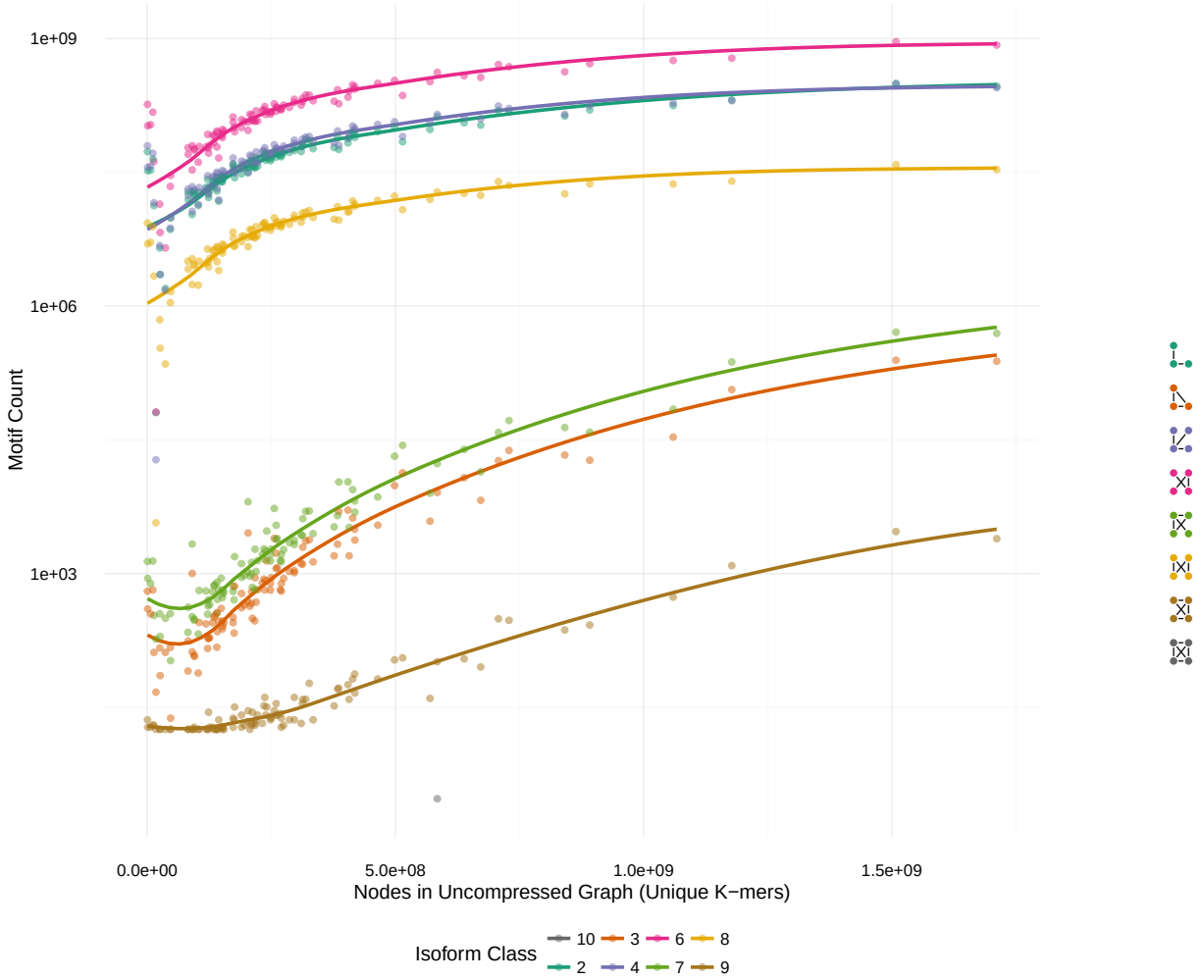


Fig. 3. Uncompressed De Bruiin Graph Size versus number of motifs in the graph for each isoform class across all 101 genomes. Each genome has a point for each motif.

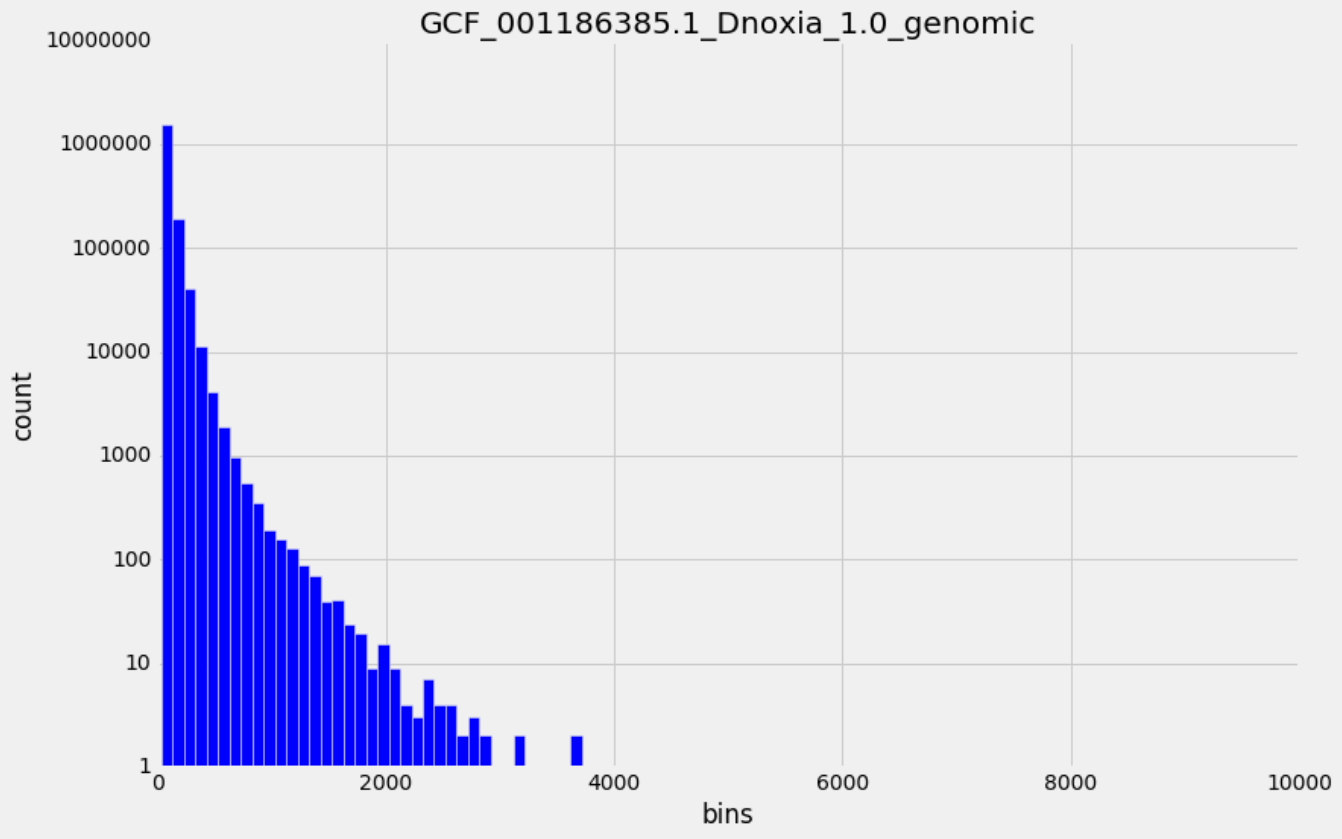


Fig. 4. Histogram for the length of linear paths in the Diuraphis noxia (Russian wheat aphid) genome. Most linear paths are not much longer than $k = 31$, but it still leads to a big decrease in node cardinality in the compressed graph

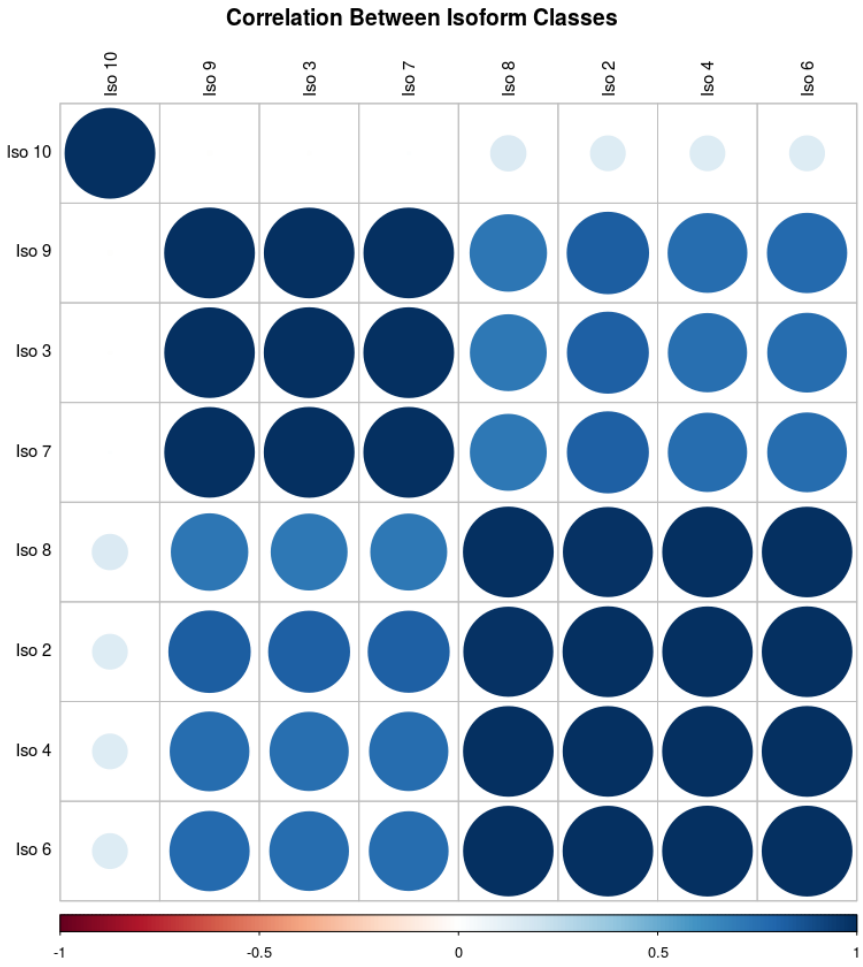


Fig. 5. Correlations between motif counts in all genomes. Motifs which are each others subgraphs are highly correlated.

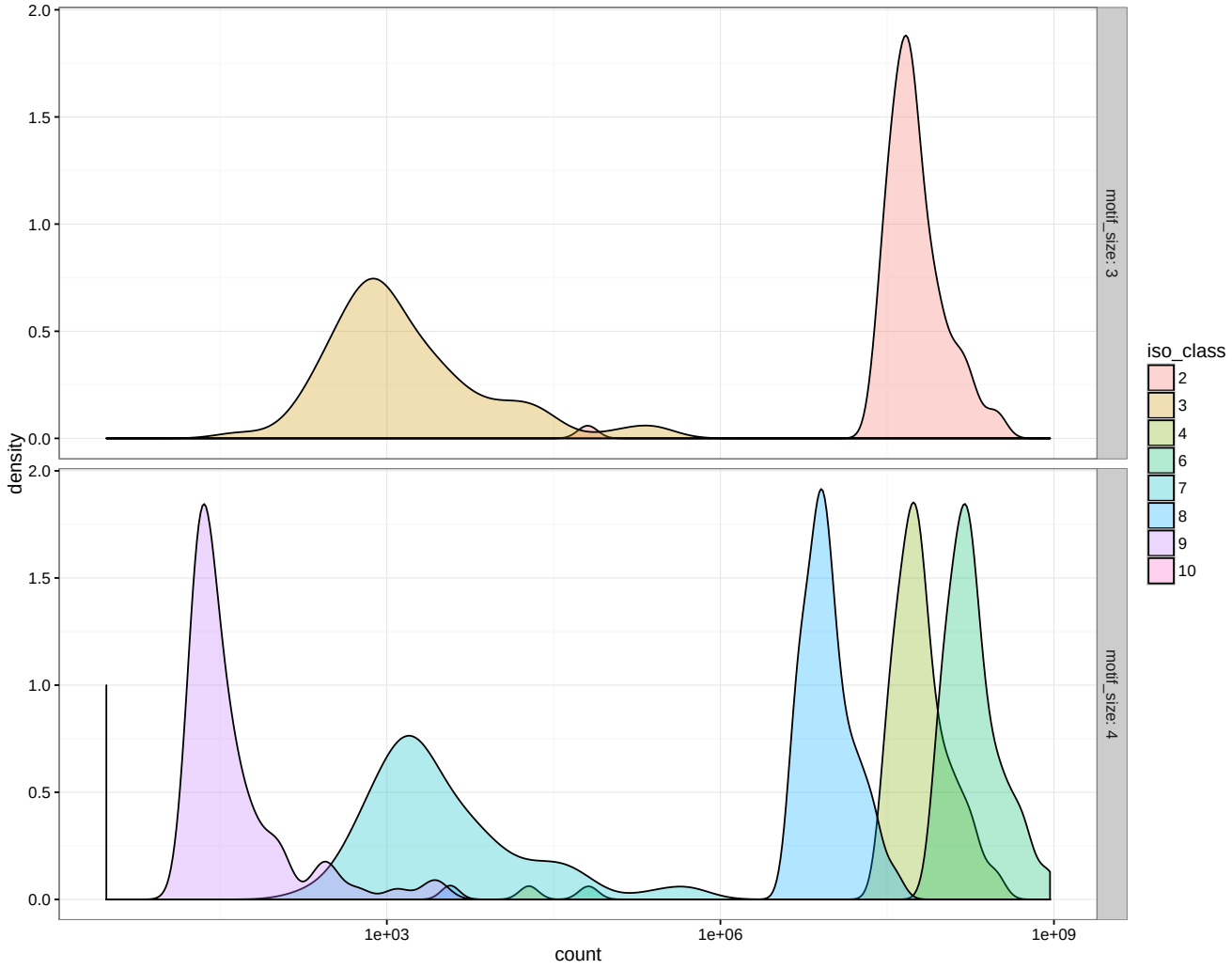


Fig. 6. Kernel density estimates of motif counts for each motif.