HHS Public Access

Author manuscript

J Comput Biol. Author manuscript; available in PMC 2018 January 24.

Published in final edited form as:

J Comput Biol. 2017 November ; 24(11): 1071–1080. doi:10.1089/cmb.2017.0013.

U_{50}: A New Metric for Measuring Assembly Output Based on Non-Overlapping, Target-Specific Contigs

Christina J. Castro[1] and **Terry Fei Fan Ng**[2]

[1]Oak Ridge Institute for Science and Education, Oak Ridge, Tennessee

[2]Division of Viral Diseases, Centers for Disease Control and Prevention, Atlanta, Georgia

Abstract

Advances in next-generation sequencing technologies enable routine genome sequencing, generating millions of short reads. A crucial step for full genome analysis is the de novo assembly, and currently, performance of different assembly methods is measured by a metric called N_{50}. However, the N_{50} value can produce skewed, inaccurate results when complex data are analyzed, especially for viral and microbial datasets. To provide a better assessment of assembly output, we developed a new metric called U_{50}. The U_{50} identifies unique, target-specific contigs by using a reference genome as baseline, aiming at circumventing some limitations that are inherent to the N_{50} metric. Specifically, the U_{50} program removes overlapping sequence of multiple contigs by utilizing a mask array, so the performance of the assembly is only measured by unique contigs. We compared simulated and real datasets by using U_{50} and N_{50}, and our results demonstrated that U_{50} has the following advantages over N_{50}: (1) reducing erroneously large N_{50} values due to a poor assembly, (2) eliminating overinflated N_{50} values caused by large measurements from overlapping contigs, (3) eliminating diminished N_{50} values caused by an abundance of small contigs, and (4) allowing comparisons across different platforms or samples based on the new percentage-based metric UG_{50}%. The use of the U_{50} metric allows for a more accurate measure of assembly performance by analyzing only the unique, non-overlapping contigs. In addition, most viral and microbial sequencing have high background noise (i.e., host and other non-targets), which contributes to having a skewed, misrepresented N_{50} value—this is corrected by U_{50}. Also, the UG_{50}% can be used to compare assembly results from different samples or studies, the cross-comparisons of which cannot be performed with N_{50}.

Keywords

genome assembly; N_{50}; next-generation sequencing; U_{50}

Address correspondence to: Dr. Terry Fei Fan Ng, Division of Viral Diseases, Centers for Disease Control and Prevention, 1600 Clifton Road NE, Mailstop G-10, Atlanta, GA 30329, ylz9@cdc.gov.

AUTHORS' CONTRIBUTION

Both authors designed and performed research, analyzed data, and wrote and approved the final article.

AUTHOR DISCLOSURE STATEMENT

No competing financial interests exist.

1. INTRODUCTION

Next-Generation Sequencing (NGS) is becoming the laboratory standard for genome sequencing, pathogen discovery, and advanced molecular detection. NGS generates a tremendous amount of short sequence reads, and one of the most common ways to analyze these data is by *de novo* assembly. Unfortunately, genome assembly remains a very difficult problem, which is made more challenging by shorter reads, non-uniform coverage of the target, and unreliable long-range linking information (Miller et al. 2010). Assemblies are measured by the size and accuracy of their contigs and scaffolds (Miller et al. 2010). Currently, the performance of a *de novo* assembly is measured by a metric called N_{50}.

The N_{50} value is a measurement of the assembly quality of NGS data by determining how well an assembler performs in forming contigs and scaffolds. N_{50} is defined as a weighted median statistic such that 50% of the entire assembly is contained in contigs that are equal to or larger than this value. Though assembly accuracy is extremely hard to measure, the N_{50} value has thus far been the most common metric to use for genomic assembly completeness. Other metrics can be used in determining overall assembly performance, but they are all based on the N_{50} statistic. Generally, it is assumed that the higher the N_{50} value, the more accurate the assembly.

Although calculation of the N_{50} is a common practice among studies analyzing NGS datasets, the N_{50} value has several major disadvantages that can sometimes produce inaccurate results (Scott 2014). First, a poor assembly can force unrelated reads and contigs into supercontigs, resulting in an erroneously large N_{50}. Second, using N_{50} and NG_{50} (see Table 1 for definitions) for metagenomics, microbial or viral datasets are problematic because a small fraction of the reads are of the targeted genome (Naccache et al. 2014) and because of the background reads (mostly of cellular origin) skewing the results. Third, N_{50} does not account for resulting contigs that are non-unique or overlapping; these overlapping contigs can sometimes greatly inflate the N_{50} value and, hence, miscalculate true performance of the NGS assembly. Finally, comparing all assemblies based solely on the N_{50} is impossible, because N_{50} is a number-based metric that is calculated from total contig length and it does not allow a fair comparison across different platforms or samples (Miller et al. 2010). Therefore, an alternative formula is needed.

This article describes a new metric called U_{50}, as well as a computer algorithm that can calculate U_{50} automatically for any NGS data, for applications mainly targeting viral and microbial datasets. Using contig sequences that are generated by an assembly program as input, the U_{50} algorithm identifies unique regions from those contigs, then applies a brute force, cumulative sum method to calculate the U_{50} metric. Our U_{50} metric aims at circumventing the limitations of N_{50} by identifying unique, target-specific contigs by using a reference sequence as baseline.

2. IMPLEMENTATION

2.1. Definitions

See Table 1.

2.2. Algorithm

The U_{50} program requires Python to be installed. Two input files are needed from the user: a multi-FASTA file containing all the contigs and a FASTA file including only the reference genome. First, the user can run the provided bash shell script to convert the two input files into a sorted BED file, using Bowtie2, SAMtools, and BEDtools (Table 2). The U_{50} script will then compare the sorted BED file listing all the contig coordinates to the reference genome, using the U_{50} calculation and mask array as described later. The Quality Assessment Tool for Genome Assemblies (QUAST) program is not necessary for the U_{50} program to execute, but the added output information from QUAST helps make informed decisions about the quality of the assembly when combined with the U_{50} output data.

During execution, the U_{50} program first sorts contigs by their lengths, starting from the longest to the shortest. Starting with the longest contig, the start and stop coordinates are used to compare it with the reference sequence at those specific coordinate positions. The coordinates are used to align the contig to the reference and find overlapping regions. If an overlap is identified, then those coordinates are removed and only the coordinates that reflect a unique contig are retained. To maintain order and position of all contigs, a mask array is used.

2.3. Using a mask array to determine unique regions

We constructed a mask array (Fig. 1) to be used as a switch to find overlapping regions. The length of the mask array was assigned to be the length of the input reference sequence. The mask array acts as a running tally that keeps track of the unique coordinates, where “1” denotes coordinates that already have a mapped contig, and “0” denotes coordinates that have yet to be covered. The initial mask array was assigned all zeros in memory, because no contigs have been mapped yet.

Starting from the first contig in the sorted list, the contig coordinate positions directly align with the index positions of the mask array. This is because both the BED file and the array have a zero-based counting system.

When the mask array is compared with each of the contigs in order, the contig coordinates are first aligned with the mask array index. When a contig coordinate occupies a mask array index containing “0” in its current stage, the contig coordinate is denoted as a unique region; then, the mask array is updated as “1” in this coordinate. On the other hand, when a mask array index contains a “1,” the contig coordinate is discarded as an overlapping region. After performing all contig alignments, if a “0” is still present in the mask array, this denotes a gap in coverage. All unique contigs are then retained for the U_{50} calculation.

The same calculation used for the N_{50} is performed on the unique contigs, thus producing the U_{50} value. A brute force, cumulative sum method is used to find the N_{50}, NG_{50}, L_{50}, U_{50}, UG_{50}, and UL_{50} values.

2.4. N_{50} calculation

Step 1: The first step for calculating N_{50} involves ordering contigs by their lengths from the longest (c_1) to the shortest (c_n).

$$c_1,\ c_2,\ c_3,\ c_4,\ c_5,\ \ldots,\ c_n$$

Step 2: Calculate the cutoff value by summing all contigs and multiplying by the threshold percentage (x); examples include: N_{25}, N_{50}, N_{75}, N_{90}, etc. This example is for N_{50}, so the threshold percentage is 50%.

$$\mathrm{N}_x \text{ cutoff} = \left(\sum_{k=1}^{n} c_k\right) * x\%$$

that is, for N_{50}

$$\mathrm{N}_{50} \text{ cutoff} = \left(\sum_{k=1}^{n} c_k\right) * 50\%$$

Step 3: Starting from the longest contig, the lengths of each contig are summed, until this running sum is greater than or equal to the N_x cutoff (e.g., for N_{50}, it would be 50% of the total length of all contigs in the assembly).

Step 4: The N_{50} of the assembly is the length of the shortest contig at the first instance where the running sum becomes greater than or equal to the N_{50} cutoff.

$$\mathrm{N}_{50} = c_k, \text{ where minimum } \left(\sum_{k=1}^{L} c_k\right) \geq \mathrm{N}_{50} \text{ cutoff}$$

whereas L_{50} = the length of contig number L

NG_{50} follows the same steps except that Step 2 is modified to the following:

$$\mathrm{NG}_{50} \text{ cutoff} = (\text{Length of Reference Genome}) * 50\%$$

2.5. U_{50} calculation

Our U_{50} metric aims at circumventing the limitations of N_{50} by identifying unique, target-specific contigs by using a reference sequence as baseline.

Step 1: The first step for calculating U_{50} involves ordering contigs by their lengths from the longest (c_1) to the shortest (c_n).

$$c_1,\ c_2,\ c_3,\ c_4,\ c_5,\ \ldots,\ c_n$$

Step 2: All contigs are mapped to a reference genome, starting with c_1. Using the mask array, only the unique portions of each contig are preserved (c'), and all other regions and

non-mapping contigs are removed. Then, these modified contigs are sorted by length from the longest (c'_1) to the shortest (c'_n).

$$c'_1, c'_2, c'_3, c'_4, c'_5, \ldots, c'_n$$

Step 3: From the modified list of contigs, the summation of all contigs is found and multiplied by the threshold percentage. This example is for U_{50}, so the threshold percentage is 50%.

$$\mathrm{U}_{50}\ \mathrm{cutoff} = \left(\sum_{k=1}^{n} c'_k\right) * 50\%$$

c' = modified contigs with overlapping regions removed, and all non-target contigs removed.

Step 4: Starting from the longest contig, the lengths of each contig are summed, until this running sum is greater than or equal to the U_{50} cutoff (50% of the total length of all contigs in the assembly).

Step 5: The U_{50} of the assembly is the length of the shortest contig at the first instance where the running sum becomes greater than or equal to the U_{50} cutoff.

$$\mathrm{U}_{50} = c'_k,\ \text{where minimum}\left(\sum_{k=1}^{L} c'_k\right) \geq U_{50}\ \mathrm{cutoff}$$

whereas UL_{50} = the length of contig number L

UG_{50} follows the same steps except:

Step 3 is modified to the following:

$$\mathrm{UG}_{50}\ \mathrm{cutoff} = (\mathrm{Length\ of\ Reference\ Genome}) * 50\%$$

UG_{50}% follows the same steps as UG_{50} with one additional step:

Step 6 involves calculating the UG_{50}%.

$$\mathrm{UG}_{50}\% = 100 * \left(\frac{UG_{50}}{Length\ of\ reference\ genome}\right)$$

The output contains four files in total: AssemblyStatistics.txt, contigs.txt, gaps.txt, and overlaps.txt. The AssemblyStatistics.txt file will show the standard output that prints to the terminal screen. The contigs.txt file shows the contig coordinates and a count of the unique base positions that a particular contig covers. The overlaps.txt file shows the index position

of an overlap and which contig that overlap belongs to. The gaps.txt file shows the index position of any gaps in coverage.

3. RESULTS

3.1. U_{50} implementation with theoretical datasets

We constructed six different types of theoretical assembly results (A–F), increasing in complexity (Fig. 2), to compare the values for N_{50}, NG_{50}, U_{50}, UG_{50}, and UG_{50}%. For the assemblies that generate long contigs with little overlap or gaps (e.g., A–C), N_{50} appears approximately equal to the U_{50} value. The U_{50} metric really excels in situations where assemblies generate short, fragmented contigs (e.g., E and F), skewing the N_{50} downward, or non-target-specific long contigs (e.g, D) that skew the N_{50} upward.

Figure 2A is the simplest example. When a *de novo* assembly produces one contig that is the length of the genome, then the N_{50}, NG_{50}, U_{50}, and UG_{50} all return the same values.

Figure 2B simulates an assembly producing two contigs totaling the length of the genome, with a single overlapping region. The N_{50}, NG_{50}, U_{50}, and UG_{50} values are identical, even after removing the overlapping region to calculate U_{50} and UG_{50}.

Figure 2C simulates an assembly producing multiple overlapping contigs totaling the length of the genome. This is often observed in earlier de Bruijn graph assemblers that produce hundreds or thousands of overlapping small contigs (Deng et al. 2015), skewing the result toward a lower N_{50}. By removing the short overlaps, the U_{50} estimate corrects for the underestimation caused by the small contigs.

Figure 2D simulates a scenario where the final assembly contains long non-targeted sequences. This is exemplified by sequencing bacterial and/or viral genomes with a high background of host cellular nucleic acid. In this example, the N_{50} is overinflated, demonstrated by having an N_{50} that is much longer than the targeted genome. The U_{50} is a better reflection of the actual assembly performance.

Figure 2E simulates a scenario where the final assembly contains numerous small contigs. The over-abundance of these small contigs generally skews the N_{50} toward the smaller contig size. When removing the overlapping regions, the U_{50} is a better representation of the assembly performance.

Figure 2F simulates the most realistic example of a *de novo* assembly's contig output. There are duplicated contigs, overlapping regions, and a gap region. There are varying sizes of contigs, and many contain overlapping regions, which skew the N_{50} downward. Once these duplicated contigs and overlapping regions are removed, the U_{50} provides a slightly better representation of the assembly performance.

Generally, it is assumed that the higher the N_{50}, the better the assembly. However, it is important to keep in mind that a poor assembly that has forced unrelated reads and contigs into scaffolds can have an erroneously large N_{50}. Using the U_{50} metric to discount any reads

that do not belong to the given reference genome may minimize the effects of overabundant small, fragmented contigs.

3.2. U_{50} implementation with published and in-house datasets

To demonstrate the U_{50} algorithm, various published and in-house datasets were compared, including four Illumina MiSeq bacteria samples that were analyzed in the GAGE-B paper (Magoc et al. 2013), four enterovirus D68 samples, three other types of picornaviruses, one Black Queen cell virus, one hepatitis E virus, one crAssphage, and one rhabdovirus Bas-Congo.

All 15 samples were assembled by using four different *de novo* assemblers: ABySS v.1.9 (Simpson et al. 2009); SOAPdenovo2 v.r240 (Luo et al. 2012); SPAdes v. 3.6.2 (Bankevich et al. 2012; Nurk et al. 2013); and Velvet v.1.2.10 (Zerbino and Birney 2008). Only the assembly output contig files were available from the four bacteria samples from the GAGE-B dataset, so the same approach was used for the other 11 samples. The UG_{50}% was used to evaluate and compare samples across all assemblers. Our results show that the SPAdes assembler consistently generates a higher UG_{50}% compared with the other three assemblers, suggesting that the SPAdes assembler generates the longest contigs and outperforms the others.

For most viral samples, the UG_{50}% performance metric for SPAdes is consistently more than 90%, showing that it can produce full or near full genomes. In contrast, the UG_{50}% performance metric for Velvet is always lower than 20%, indicating that the largest contig would not be more than one fifth of the genome, similar to previous findings (Magoc et al. 2013). For SOAPdenovo2 and ABySS, the performance varied from sample to sample, with the UG_{50}% ranging from <1% to 100%. For the bacterial samples, a similar trend is apparent. The UG_{50}% for SPAdes is higher than 12%, whereas the UG_{50}% for the other three assemblers is lower than 5%.

For many samples, the U_{50} is within 10% of the N_{50}. SOAPdenovo2 and Velvet have the largest differences when comparing the N_{50} and U_{50} values, because these assemblers produce many small contigs, as portrayed in Figure 2E. This typically results in the N_{50} value being skewed toward the smaller contig size. By removing the overlapping regions, the U_{50} value is much higher and would be a more accurate estimate (see picornaviruses SOAPdenovo2 and Velvet results for EV-C105 and EV-C117 in Figure 3). In these cases, the U_{50} is a better estimate compared with the N_{50} for SOAPdenovo2 and Velvet because of removal of the non-target specific contigs.

When the assembler can generate long contigs covering almost the entire length of the reference genome, then the N_{50} and the U_{50} values are identical (see examples Figure 2A and B in the simulated data and the SPAdes results for EV-D68, EV-C105, and EV-C117). For the remaining of the SPAdes results, the N_{50} and U_{50} values are very close. The slight difference is most likely due to the removal of the duplicated and overlapping reads.

Noticeably, the NG_{50} value calculated for all Black Queen virus assemblies, hepatitis E virus assemblies, and the SPAdes assembly for crAssphage returned values that are longer than

the reference genome. This is because NG_{50} estimates the reference genome length based on the input contigs themselves (Ghodsi et al. 2013), not on an input reference genome. This exemplifies that the overinflated NG_{50} results can be inaccurate and misleading.

3.3. Availability of data and material

The datasets generated during and/or analyzed during the current study and the U_{50} python script are available in the U_{50} Github repository, https://github.com/CDCgov/U50.

4. DISCUSSION AND CONCLUSIONS

In this study, we describe the U_{50} metric as a tool to evaluate assembly performance, aiming at circumventing some limitations that are inherent to the N_{50} metric. The core spirit of U_{50} is in removing noise and finding those unique regions that align to a targeted reference genome. Major advantages include the following: (1) reducing erroneously large N_{50} values due to a poor assembly, (2) eliminating overinflated N_{50} values caused by large measurements from overlapping contigs, (3) eliminating diminished N_{50} values caused by an abundance of small contigs, and (4) allowing comparisons across different platforms or samples based on the percentage-based metric UG_{50}%.

The U_{50} assembly metric will be particularly useful for viral and microbial sequencing of samples with high background noise. This “needle-in-a-haystack” problem proves cumbersome when trying to assemble such small viral contigs into complete genomes with exponentially larger non-target contigs. The reads often do not overlap sufficiently to allow the *de novo* assembler to properly form long contigs (Kostic et al. 2011), and teasing out the true viral contigs from the rest is challenging. The U_{50} metric allows for only the proper viral contigs to be used when calculating the assembly performance, as opposed to using all of the contigs.

Despite the advantages, there are some limitations that prevent usage of the U_{50} under all circumstances. First, U_{50} can only be calculated if a reference is available. Therefore, this is not a metric to use during the assembling of unique prototype genomes. Second, because the overlapping regions are removed for the calculation, U_{50} does not account for coverage of the genome.

Since only unique, non-overlapping contigs are used in the calculation of U_{50}, the sum of the unique contig length is usually less than the sum of the original contigs. In the instance where all the unique contigs do not sum to the percentage threshold of 50%, the U_{50} program returns a “0.” This is indicative of the input contigs not covering 50% of the reference genome. In such cases, we recommend using a lower threshold such as U_{10} or U_{25}. These will typically return a result, but the overall assembly when aligned to a reference is still poor.

The U_{50} assembly metric is a tool to be used in conjunction with the commonly used N_{50}, L_{50}, and NG_{50} metrics. When used in unison, it gives a clearer picture of the performance and accuracy of the overall assembly. The UG_{50}%, as a percentage-based metric, can be used to compare assembly results from different samples or studies. The use of this new

metric and its software could facilitate a better comparison of assemblies, especially with viral and microbial datasets.

Acknowledgments

The authors thank Steve Oberste, Paul Rota, Greg Doho, Roman Tatusov, and Edward Ramos for their constructive comments. This work was supported by Federal appropriations to the Centers for Disease Control and Prevention (CDC), through the Advanced Molecular Detection Initiative line item. This research was also supported in part by an appointment to the Research Participation Program at the CDC administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S Department of Energy and CDC.

References

Bankevich A, Nurk S, Antipov D, et al. SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. J Comput Biol. 2012; 19:455–477. [PubMed: 22506599]

Cock PJA, Antao T, Chang JT, et al. Biopython: Freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. 2009; 25:1422–1423. [PubMed: 19304878]

Deng X, Naccache SN, Ng T, et al. An ensemble strategy that significantly improves de novo assembly of microbial genomes from metagenomic next-generation sequencing data. Nucl Acids Res. 2015; 43:e46–e46. [PubMed: 25586223]

Ghodsi M, Hill CM, Astrovskaya I, et al. De novo likelihood-based measures for comparing genome assemblies. BMC Res Notes. 2013; 6:1–18. [PubMed: 23281703]

Gurevich A, Saveliev V, Vyahhi N, et al. QUAST: quality assessment tool for genome assemblies. Bioinformatics. 2013; 29:1072–1075. [PubMed: 23422339]

Kostic AD, Ojesina AI, Pedamallu CS, et al. PathSeq: A comprehensive computational tool for the identification or discovery of microorganisms by deep sequencing of human tissue. Nat Biotechnol. 2011; 29:393–396. [PubMed: 21552235]

Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012; 9:357–359. [PubMed: 22388286]

Li H. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. Bioinformatics. 2011; 27:2987–2993. [PubMed: 21903627]

Li H, Handsaker B, Wysoker A, et al. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009; 25:2078–2079. [PubMed: 19505943]

Luo R, Liu B, Xie Y, et al. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience. 2012; 1:1–6. [PubMed: 23587310]

Lutz, M. Learning Python. 5th. O'Reilly Media; Sebastobol, CA: 2013.

Magoc T, Pabinger S, Canzar S, et al. GAGE-B: an evaluation of genome assemblers for bacterial organisms. Bioinformatics (Oxford, England). 2013; 29:1718–1725.

Miller JR, Koren S, Sutton G. Assembly algorithms for next-generation sequencing data. Genomics. 2010; 95:315–327. [PubMed: 20211242]

Naccache SN, Federman S, Veeraraghavan N, et al. A cloud-compatible bioinformatics pipeline for ultra-rapid pathogen identification from next-generation sequencing of clinical samples. Genome Res. 2014; 24:1180–1192. [PubMed: 24899342]

Nurk, S., Bankevich, A., Antipov, D., et al. Assembling genomes and mini-metagenomes from highly chimeric reads, 158–170. In: Deng, M.Jiang, R.Sun, F., Zhang, X., editors. Research in Computational Molecular Biology: 17th Annual International Conference, RECOMB 2013, Beijing, China, April 7–10, 2013. Springer Berlin Heidelberg; Berlin, Heidelberg: 2013.

Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 2010; 26:841–842. [PubMed: 20110278]

Scott, DC. Utilizing Next Generation Sequencing to Generate Bacterial Genomic Sequences for Evolutionary Analysis. Doctoral dissertation. 2014. Retrieved from http://scholarcommons.sc.edu/etd/2887

Simpson JT, Wong K, Jackman SD, et al. ABySS: A parallel assembler for short read sequence data. Genome Res. 2009; 19:1117–1123. [PubMed: 19251739]

Zerbino DR, Birney E. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Genome Res. 2008; 18:821–829. [PubMed: 18349386]

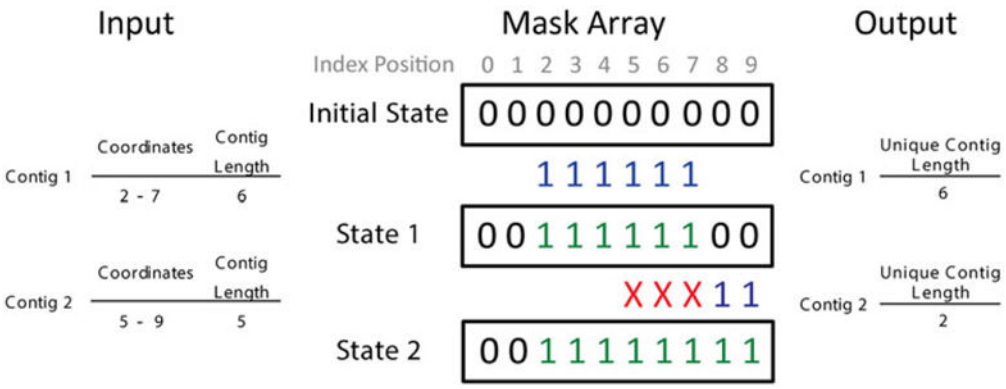


FIG. 1.

The use of mask array in U_{50}. A schematic diagram demonstrating the use of a mask array to identify the unique contig length for two contigs.

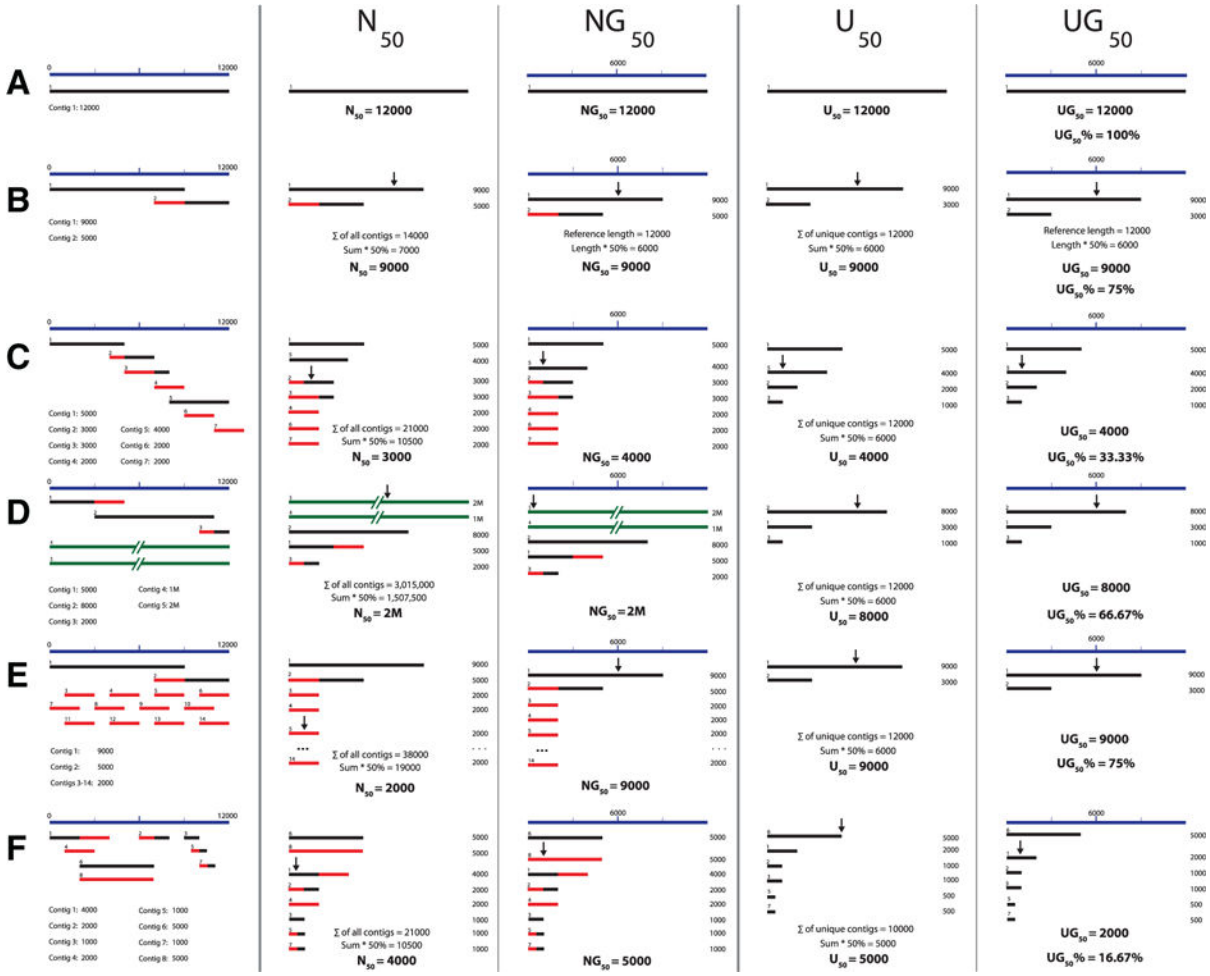
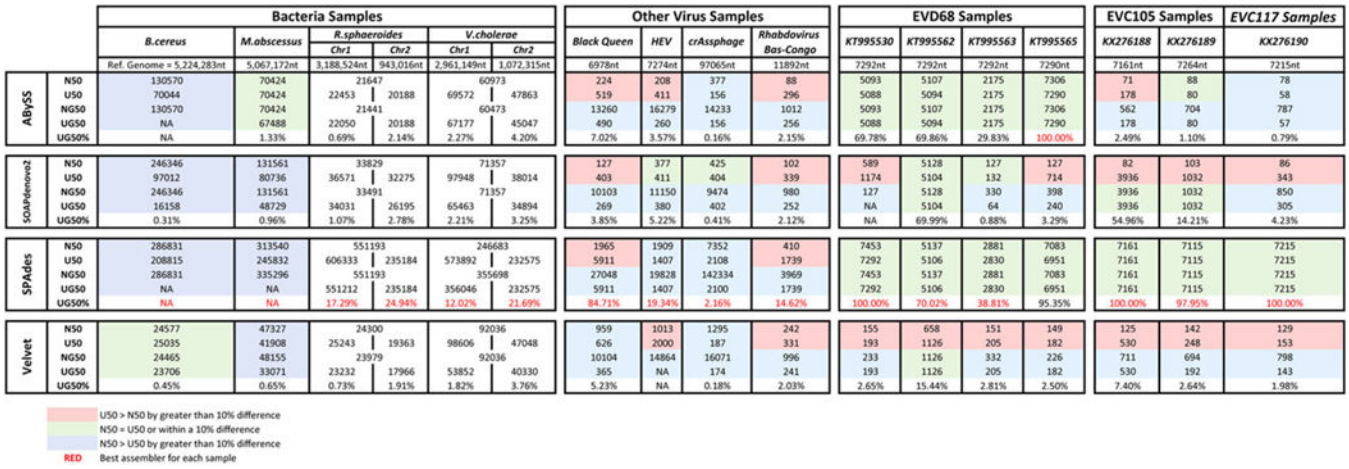


FIG. 2.

Simulated output of different assemblies, with the N_{50}, NG_{50}, U_{50}, UG_{50}, and UG_{50}% values calculated. Blue denotes the reference sequence, black denotes a unique contig, red denotes an overlapping region of a contig, and green denotes a non-target contig.



		Bacteria Samples						Other Virus Samples				EVD68 Samples				EVC105 Samples		EVC117 Samples
		B.cereus	M.abscessus	R.sphaeroides		V.cholerae		Black Queen	HEV	crAssphage	Rhabdovirus Bas-Congo	KT995530	KT995562	KT995563	KT995565	KX276188	KX276189	KX276190
		Ref. Genome = 5,224,283nt	5,067,172nt	3,188,524nt	943,016nt	2,961,149nt	1,072,315nt	6978nt	7274nt	97065nt	11892nt	7292nt	7292nt	7292nt	7290nt	7161nt	7264nt	7215nt
ABySS	N50	130570	70424	21647		60973		224	208	377	88	5093	5107	2175	7306	71	88	78
	U50	70044	70424	22453	20188	69572	47863	519	411	156	296	5088	5094	2175	7290	178	80	58
	NG50	130570	70424	21441		60473		13260	16279	14233	1012	5093	5107	2175	7306	562	704	787
	UG50	NA	67488	22050	20188	67177	45047	490	260	156	256	5088	5094	2175	7290	178	80	57
	UG50%	NA	1.33%	0.69%	2.14%	2.27%	4.20%	7.02%	3.57%	0.16%	2.15%	69.78%	69.86%	29.83%	100.00%	2.49%	1.10%	0.79%
SOAPdenovo2	N50	246346	131561	33829		71357		127	377	425	102	589	5128	127	127	82	103	86
	U50	97012	80736	36571	32275	97948	38014	403	411	404	339	1174	5104	132	714	3936	1032	343
	NG50	246346	131561	33491		71357		10103	11150	9474	980	127	5128	330	398	3936	1032	850
	UG50	16158	48729	34031	26195	65463	34894	269	380	402	252	NA	5104	64	240	3936	1032	305
	UG50%	0.31%	0.96%	1.07%	2.78%	2.21%	3.25%	3.85%	5.22%	0.41%	2.12%	NA	69.99%	0.88%	3.29%	54.96%	14.21%	4.23%
SPAdes	N50	286831	313540	551193		246683		1965	1909	7352	410	7453	5137	2881	7083	7161	7115	7215
	U50	208815	245832	606333	235184	573892	232575	5911	1407	2108	1739	7292	5106	2830	6951	7161	7115	7215
	NG50	286831	335296	551193		355698		27048	19828	142334	3969	7453	5137	2881	7083	7161	7115	7215
	UG50	NA	NA	551212	235184	356046	232575	5911	1407	2100	1739	7292	5106	2830	6951	7161	7115	7215
	UG50%	NA	NA	17.29%	24.94%	12.02%	21.69%	84.71%	19.34%	2.16%	14.62%	100.00%	70.02%	38.81%	95.35%	100.00%	97.95%	100.00%
Velvet	N50	24577	47327	24300		92036		959	1013	1295	242	155	658	151	149	125	142	129
	U50	25035	41908	25243	19363	98606	47048	626	2000	187	331	193	1126	205	182	530	248	153
	NG50	24465	48155	23979		92036		10104	14864	16071	996	233	1126	332	226	711	694	798
	UG50	23706	33071	23232	17966	53852	40330	365	NA	174	241	193	1126	205	182	530	192	143
	UG50%	0.45%	0.65%	0.73%	1.91%	1.82%	3.76%	5.23%	NA	0.18%	2.03%	2.65%	15.44%	2.81%	2.50%	7.40%	2.64%	1.98%

U50 > N50 by greater than 10% difference
N50 = U50 or within a 10% difference
N50 > U50 by greater than 10% difference
RED Best assembler for each sample

FIG. 3.

Comparison of the N_{50}, U_{50}, NG_{50}, UG_{50}, and UG_{50}% for different bacterial and viral datasets using four different assemblers. Contigs for the bacteria dataset were retrievedfrom Magoc et al. (2013), whereas all other contigs were generated in-house. All datasets were sequenced by using the Illumina MiSeq.

Table 1

N_{50} and U_{50} Assembly Metric Definitions

N_{50} Metrics		**U_{50} Metrics**	
N_{50}	The length of the smallest contig such that 50% of the sum of all contigs is contained in contigs of size N_{50} or larger.	U_{50}	The length of the smallest contig such that 50% of the sum of all unique, target-specific contigs is contained in contigs of size U_{50} or larger.
L_{50} [LG_{50}]	The number of contigs whose length sum produces N_{50} [NG_{50}].	UL_{50} [ULG_{50}]	The number of contigs whose length sum produces U_{50} [UG_{50}].
NG_{50}	The length of the smallest contig such that 50% of the reference genome is contained in contigs of size NG_{50} or larger. NG_{50} estimates the genome size based on the input contig lengths, not a reference genome as input.	UG_{50}	The length of the smallest contig such that 50% of the reference genome is contained in unique, target-specific contigs of size UG_{50} or larger.
		$UG_{50}\%$ $=100 * (\frac{UG_{50}}{Length\ of\ reference\ genome})$	The estimated coverage length of the UG_{50} in direct relation to the length of the reference genome.

Table 2

Software Used in the U_{50} Package

Program/language	Version	Application	Execution order
Bowtie2	2.2.4	Maps contigs to reference, creating an SAM file (Langmead and Salzberg 2012)	1
SAMtools	1.2	Converts SAM to BAM (Li 2011; Li et al. 2009)	2
BEDTools	2.17.0	Converts BAM to BED (Quinlan and Hall 2010)	3
Python *(BioPython)*	2.7.3 1.66	Executes the U_{50} program (Cock et al. 2009; Lutz 2013)	4
QUAST	2.3	Calculates assembly metrics (N_{50}, N_{75}, L_{50}, GC%, etc.) based on the input contig file (Gurevich et al. 2013)	5