University of Glasgow | School of Life Sciences

Benchmarking a pipeline for detecting defective interfering particles from Next Generation Sequencing data

Jane Walls *Student ID: 2598002*

Supervised by Dr. Joseph Hughes

University of Glasgow | MSc Bioinformatics

A report submitted in partial fulfilment of the requirements for the MSc Bioinformatics Degree at The University of Glasgow

Summary

Introduction: Defective interfering particles (DIPs), are faulty viral genomes with large deletions or other errors which prevents the virus from functioning normally. They appear during viral replication and are theorised to be created by most viruses. Yet there is little research in this area which clearly outlines the exact effect these have in different viruses. Some studies have shown an effect on virus pathogenicity and may have the potential to be used as an antiviral. Therefore, it is imperative to understand the tools that are being used to detect DIPs. This study aims to look at the current detection tools and benchmark a method for other studies who aim to detect DIPs and create tools to facilitate this. Methods: ViReMa and DI-Tector were the tools chosen for comparison, data was simulated from the Bluetongue Virus (BTV) genome with four different methods, primarily representing deletion and copy-back type DIPs. Also, next-generation sequenced BTV samples isolated from a bovine host were analysed for DIPs. Results: Tools were created to simulate DIPs and parse outputs for better comparison. DI-Tector was found to be more accurate across all DIP types. ViReMa identified significantly more DIPs but had a high false positive rate (~99.9%). DIPs were found in all BTV samples with both tools. DIPs generally either started at the 5' end or finished at the 3' end of the transcription region, varying in deletion size. Conclusion: These tools are reliable enough to conclude that there are DIPs present in BTV, however much development is needed to create conclusions around exact break points and re-initiation sites and developing a tool that has a better balance of sensitivity to specificity.

Key Abbreviations

DIP = Defective Interfering Particle
BrP = Break Point
RI = Re-initiation Point
WT = Wild Type
NGS = Next Generation Sequencing
BTV = Bluetongue Virus
RdRp = RNA-dependent RNA polymerase
STD = Standard Deviation

Table of Contents

Introduction

Defective interfering particles (DIPs) are created by most RNA viruses and are noted to be a product of faulty viral replication. Effectively creating defective genomes where essential genes are missing. It therefore does not cause infection to the host the same way the wild type (WT) virus does (Huang and Baltimore, 1970). DIPs were discovered in the 1940s by Preben Von Magnus, where he noticed "incomplete particles" that interfered with viral replication in influenza (von Magnus, 1954). It is hypothesised that these large deletions occur due to viral RNA-dependent RNA polymerase (RdRp) ceasing RNA polymerisation at one site - donor site or break point (BrP), and then continuing at another site downstream - acceptor site or re-initiation site (RI) (Figure 1.B) (Jennings et al., 1983). A similar mechanism occurs for copy-back and snap-back DIPs, where in this case the RdRp reattaches to the nascent strand, creating a complementary end (Figure 1.C,D) (Manzoni and López, 2018). Other types of DIPs have been theorised, such as mosaic DIPs, or more complex insertion DIPs (Dimmock and Easton, 2014), however this study primarily focussed on deletion and copy-back DIPs.

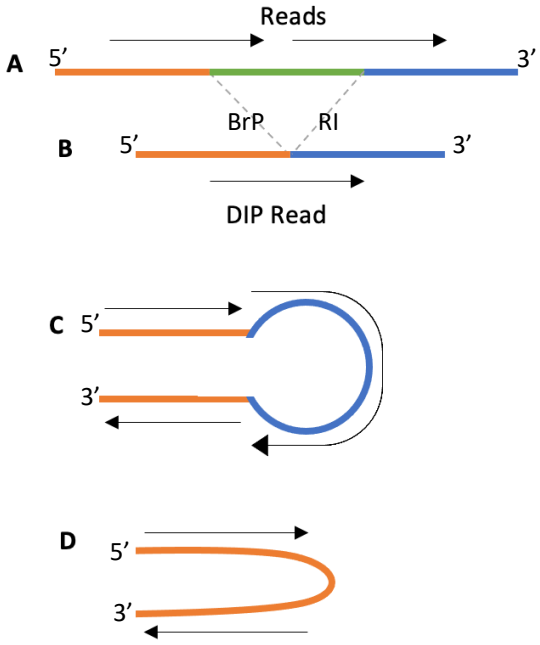


Figure 1: A) Whole viral genome B) DI Genome, C) Copy-back DIP, D) Snap-back DIP, DIP reads span over deletion sites. In whole genome reads do not span across a deletion site

Currently DIPs are poorly understood, nonetheless they are important to study, due to their "interfering" nature they have the potential to be utilised as an antiviral agent (Marriott and Dimmock, 2010). Studies found DIPs outcompete the WT virus for resources in the cell, meaning that increased numbers of DIPs showed decreased levels of WT viruses (Vignuzzi and López, 2019). Linked to this, DIPs also have been found to influence the pathogenicity of a virus. One study found a mutation in influenza A Virus that caused a lower population of DIPs, presented more severe disease outcomes than the virus that produced more DIPs (Vasilijevic et al., 2017). Many other studies have

found similar trends in other viruses such as dengue virus (Wang et al. 2020), and classical swine fever virus (Aoki et al., 2003).
Another clinical use for DIPs has thought to be live-attenuated vaccines (Ziegler and Botten, 2020). For instance, a study looking at a Lassa virus vaccine with this methodology was found to be effective in animal models (Johnson et al., 2021). Furthermore, DIPs have been tested as vaccine adjuvants, one study that tested this in Sendai-virus showed improved antibody production (Mercado-López et al., 2013). Therefore, being able to detect and study them could lead to important discoveries in viral disease control globally.

Before computer tools were developed to detect DIPs, they were detected using PCR assays, however these only find primer specific DIP species. Then Sanger technologies were used to first sequence DIPs. This was also limited as although longer reads can lead to easier identification, the low read depth was not efficient to sequence large DIP populations (Bosma et al., 2019). Later the development of Next Generation Sequencing (NGS) and tools to detect DIPs *in silico*, had the capability to sequence large DIP populations *de novo*, causing a resurge in DIP studies (Ziegler and Botten, 2020). Nevertheless, detecting these DIPs from NGS data is difficult both owing to the nature of the DIPs and the nature of NGS data (Alnaji et al., 2018). The main challenge is differentiating DIP reads from other noise such as the WT and host genetic data, and chimeras, which can all cause false positive DIPs (Beauclair et al., 2018). Additionally, potential minority haplotypes, and base calling errors in sequencing can also present challenges in identifying DIPs (Ziegler and Botten, 2020). Finally, a phenomenon called "fuzzing", this is when bases at the BrP and RI match, therefore difficult to define these exact BrP position (Routh and Johnson, 2014).
Some tools developed include; Paparazzi, which was developed in 2011 to detect both interfering small RNA and DIPs (Vodovar et al., 2011). ViReMa was developed in 2014 (Routh and Johnson, 2014). DI-Tector was developed in 2018 (Beauclair et al., 2018). DVG-Profiler, a part of the "HIVE" environment from the Food & Drug Administration in the United States of America was developed in 2019 (Bosma et al., 2019). The principal method used by detecting software in essence is to find the reads that span across BrP/RI junctions, like the read in Figure 1.B.

Benchmarking a method validates these detection processes, taking out discrepancies in the pipeline with future datasets. It can improve accuracy and reduces time for bioinformaticians before data analysis in DIPs. It also helps biologists with less bioinformatics backgrounds effectively use the tools to have the most representative results of their data. There is currently little discussion on the mentioned tools online, and at the time of writing there is no objective paper that compares these tools.
The aim of the project is to benchmark a process, for the best way to detect DIPs from NGS data and develop tools to assist with analysis. Thus, data will be simulated to test the software before using the pipeline on raw data, from which the data will then be visualised. The raw data used in this study was from the Bluetongue Virus (BTV) from a bovine host, BTV is a common RNA virus composed of 10 segments. BTV causes Bluetongue disease which infects ruminant animals. It is transmitted by various *Culicoides* insects (midges), and most commonly is fatal (Maclachlan, 2011). DIPs have only been theorised in BTV, in a study done in 1987 but not conclusively found (Cowley and Gorman, 1987).

Methods

Raw Data

Raw Data was supplied by the University of Glasgow Centre for Virus Research, which was sequenced BTV from a bovine host taken in France 2017. There were 4 blood samples (Bld_1, Bld_2, Bld_3, Bld_4), also an isolate sample (Isol) which represents the isolate of Bld_3 in *Culicoides sonorensis* (KC) cells. Samples had no PCR amplification, no ribosomal depletion, nor polyA selection. Libraries were prepared with Illumina TrueSeq Stranded mRNA kit without targeted enrichment, then were sequenced using a NextSeq500 sequencer creating paired-end data.
FastQC was used to check read length and quality (Andrews, 2010), the reads were then quality trimmed with Trim Galore at the tools default settings (Krueger, 2015). The reads were then aligned to the BTV reference with Bowtie2 (Langmead and Salzberg, 2012). An open-source script called getinstertsize.py was used to find the mean read length and

span (Wei Li, 2015). SAMtools view, sort and index were used to create and sort a bam file for each sample (Li et al., 2009).

The results from this alignment showed a mean read length of 141. 53 (standard deviation (STD) = 15.96), the mean read span was 221.19 (STD = 73.30) these results were then used as a base for the simulated data. The mean alignment rate to BTV was found to be 0.26%, therefore an alignment to the *Bos taurus* (cow) genome using the same method as above which in the blood samples had a mean alignment rate of 95.87%, and 6.48% in the Isol sample. As a result, for later analysis all the unmapped reads for the cow alignment were extracted using SAMtools fastq (Li et al., 2009).

BTV Data Simulation

Data was simulated with 4 different methods based on DIP simulation methods previous papers had used: 3 across single segments, 1 across multiple segments, and a control method that creates WT reads. A command-line program package was created called “DI Generator” (DIG) in python3 using BioPython packages (Cock et al., 2009). DIG is available with all documentation at (https://github.com/janewalls/DIG). All methods created DIPs by using the BioPython Seq package to read the chosen reference genome segment (method dependent) and wrote fasta sequences using the SeqRecord and SeqIO packages.

The methods used were as follows: “MBP” method, which stands for middle BrP, this was based off the simulation method used in the ViReMa paper (Routh and Johnson, 2014). Here the BrP was directly in the middle of the user set DIP read length. The program found two random numbers from (length ÷ 2) to the (length of the segment – (length ÷ 2)) as the BrP and RI. The read was taken from the (BrP - (length ÷ 2)) to the BrP, then from the RI to (RI + (length ÷ 2)), the sequences were concatenated to create the DIP read.

The “INDEL” method was based on a method in the DI-Tecor paper (Beauclair et al., 2018). This was similar to the “MBP” method, but the BrP occurs at a random point in the DIP. To do this a random number (j) was generated between 1 and the set read length, j was used as the length of the first portion of sequence until the BrP, and (set length – j) was used for the remainder of the DIP sequence after the RI.

2598002

The “Copyback” method was based off another simulation in DI-Tector (Beauclair et al., 2018). Here there are 4 different types of DIPs created; 5’ copy-back – where reads start at the 5’ end until the BrP, then reinitiates at a different location for a length, then reverse copies the first section starting from the BrP (Figure 1.C). 5’ snap-back – where the read will break and snap back to reverse copy the strand (Figure 1.D). 3’ copy-back, and 3’ snap-back – these are the same as their 5’ counterparts but starting on the 3’ end. Users choose the proportion of reads they want for each copy-back/snap-back type in a comma separated ratio (Table 1). For 5’ copy-back DIPs a BrP, RI and j were generated, and the reads spanned from (BrP – (j/2)) to BrP, then from RI to (RI + (read length – j)), and finally reverse copied the BrP to (BrP – (j/2)). 5’ snap-backs were made creating a read from (BrP – read length) to BrP, which was reverse copied. 3’ copy-back and snap-backs were created the same way but inverted starting from 3’ end.

“MultiSeg” and “MultiSeg2” methods are both based on a simulation method used in the paper that studied DIPs in influenza (Alnaji et al., 2019). The “MultiSeg” method was designed to more directly match the method used in the study. It created longer DIPs over two segments, which were subsequently fragmented. Although, there was no traceability of reads which containing the BrP, so was not used in analysis. “MultiSeg2” was then developed similar to “INDEL”, however the first portion of the DIP until the BrP occurred within a chosen window of nucleotides from the start of a random segment, and the RI in a window from the end of another random segment.

“NoDIP” was created to make control WT reads, which are fragments of the chosen genome segment at a given length, this allowed calculation of a true negative.

Simulation Method	*Number of simulated DIPs*	*Number of simulated reads*	*Max read length (nt)*	*Other flags used*
MBP	100,000	1,000	150	--seg 1
INDEL	100,000	1,000	150	--seg 10
Copyback	100,000	1,000	150	-c (0.45,0.05,0.45,0.05), --seg 8
MultiSeg2	100,000	1,000	150	-m 150 -t 100000 -w 600
NoDIP	100,000	1,000	150	-m 150 -t 100000 --seg 5

Table 1: Table to show methods run with flags used for each of the DIP simulation methods in DIG.py. nt = nucleotides, seg = segment, c = ratios of copy-back and snap-backs, m = max length, t = total reads, w = window for DIP creation

All random numbers were generated using the random package in python. All packages produced a collection of fasta reads in a fasta file output, and a summary file naming the DIP information (e.g. BrP and RI).
Table 1 shows methods and flags used to make the simulated data; random segments were chosen for each method, ensuring longer and shorter segments were included. Lengths were based off the FastQC and getinsersize.py analysis on raw BTV reads. Finally, /usr/bin/time was also used to get information of time, and memory usage on the tools on each method of simulation.

DIP Detectors

Paparazzi was excluded as the lead of the paper who published the code was contacted and informed the code used methods that had not been updated rendering the program unusable. Also, after investigating DVG-Profiler further, there appeared to be no installation instructions for DVG-Profiler or HIVE, therefore was excluded too. Consequently, ViReMa, and DI-Tector were the 2 programs used for comparison.
DI-Tector version 0.6 was used, which was the most recent version available at the time, and ViReMa version 0.6 was used. A more recent version of ViReMa was tested (version 0.25), but not used due to unresolvable key errors relating to the version of bwa aligner, and no documentation on the necessary bwa version required.

ViReMa, and DI-Tector were both run individually on the simulated data fasta files, using the fasta option. Parameters were chosen on each tool in the effort to match as much as possible for optimal comparison. ViReMa was set to use bwa aligner rather than the default which was Bowtie. DI-Tector used a flag to ensure all DIPs were shown, even if only observed once.
For the raw data the fastq files with the extracted BTV data (cow sequences filtered out), were run through ViReMa and DI-Tector both using the same flags as the simulated data. The extracted BTV data meant the paired ends were already joined (neither tools could account for paired end data). No host genome was given to the tool, as these reads had already been extracted. The flags used that were different included; for DI-Tector all intermediate files were kept, in ViReMa a bed file was created, and both without the fasta file flag. The Isol sample was run with the same flags as above, however in ViReMa

without the bed file output, due to time constraints and the length of time the program took to run.

Comparison Programs

A command line runnable script called Outparse.py was created, documentation can be found at (https://github.com/janewalls/DIG). This program creates a summary output that can compare ViReMa and DI-Tector to the data simulated created by DIG.py or compare the tools between themselves. The output was a parser_output.csv file, which notes: the individual DIPs with their BrP, RI and the respective segments those occur in, simulated data count (if added), and the count of DIPs that both tools found. Also, a summary file was created, showing the total DIPs found, totals matched between simulated data and tools, or between tools. A flag was created to allow for fuzzing meaning DIPs found could be within a chosen STD for the BrP and RI positions.

CompDIP.py was created to compile multiple sample results into one summary file. It took the parser_output.csv comparison between ViReMa and DI-Tector, for up to five samples. A flag was also added to filter out reads with low occurrences. It gave a table output which showed DIP counts across all samples.

OutParse.py was run on both simulated data tool outputs twice, once with STD = 0, and again with STD = 2, this was to test if fuzzing made a difference on results. Raw sample tool outputs were run with a STD = 1 to allow for some fuzzing between tools.

CompDIP.py was run twice on the 4 blood samples for comparison; once with cut off = 0, and once with cut off = 1.

	Simulated DIPs	*WT Reads*
DIPs Identified	True Positive: DIP method: Total matched DIPs	False Positive: DIP method: Total unmatched DIPs
No DIPs identified	False Negative: DIP method: Total sim – DIPs found	True Negative: “NoDIP” method: Total sim – DIPs found

Table 2: Table to show how the True Positive, False Positive, False Negative, and True Negative were calculated, which were then used for calculating sensitivity, specificity, and accuracy. “DIP Method” refers to a DIP type of simulation, e.g. “MBP” or “INDEL”

Analysis

R Studio was used to create bar charts, scatter plots (with ggplot2), and Venn diagrams (with VennDiagram) and complete other calculations for the results such as sensitivity, specificity, and accuracy. Integrated Genomics Viewer (IGV) used bed files to see junctions across single segments (Thorvaldsdottir et al., 2013). ViReMa's bed files included a high number of reads, therefore using R were filtered for DIPs that occurred more than 5 times, reducing the noise of BrP/RI junctions.

DI-Tector does not produce a bed file, so ditToBed.py was created. It took the DI-Tector_output.txt and converted the information into the BED format. Documentation for ditToBed.py can be found at (https://github.com/janewalls/DIG), in the program copy-back/snap-back DIPs were noted to be on the positive strand. The DI-Tector bed file was not filtered before entering IGV as few DIPs were found by DI-Tector overall, therefore no noise reduction was necessary.

Circos plots were generated using the circos.ca online tool, from contingency tables were generated in R from the Outparse.py output. These plots showed the relationships between DIPs that span across different segments (Krzywinski et al., 2009).

Results

Tool Output Comparison

ViReMa and DI-Tector provide varying information in their respective output files, neither are well documented on how to clearly interpret these files. DI-Tector does provide a brief description in the paper on the main output but not on the intermediate files. ViReMa does not provide any information on how to interpret their output files, so for this study they have been interpreted as follows; "ViReMa.txt" had the full path for the program and input files, "Single_Alignments.txt" listed reads that align to one area, "Unmappedreads.txt" noted reads that could not be mapped to the reference genome, "Virus_Insertions.txt" noted reads that aligned with an extra insertion noting the sequence of the insertion, "Virus_Substitutions.txt" noted any substitutions in the reads alignment, "Unknown_Recombinations.txt" noted reads that could not be

mapped to the reference or where mapping was ambiguous, and "Virus_Recombination_Results.txt" noted BrP and RI across their respective segment and the read count.

"DI-tector_output_sorted.txt" and "Virus_Recombination_Results.txt" were chosen to compare the performance of the tools. As these both contained information regarding the segment, BrP and RI. ViReMa, had segment specific information whether on the reverse strand or not, but DI-Tector did not. DI-Tector contained information regarding the specific type of DIP for each read, however ViReMa did not. Therefore, these were not considered by Outparse.py.

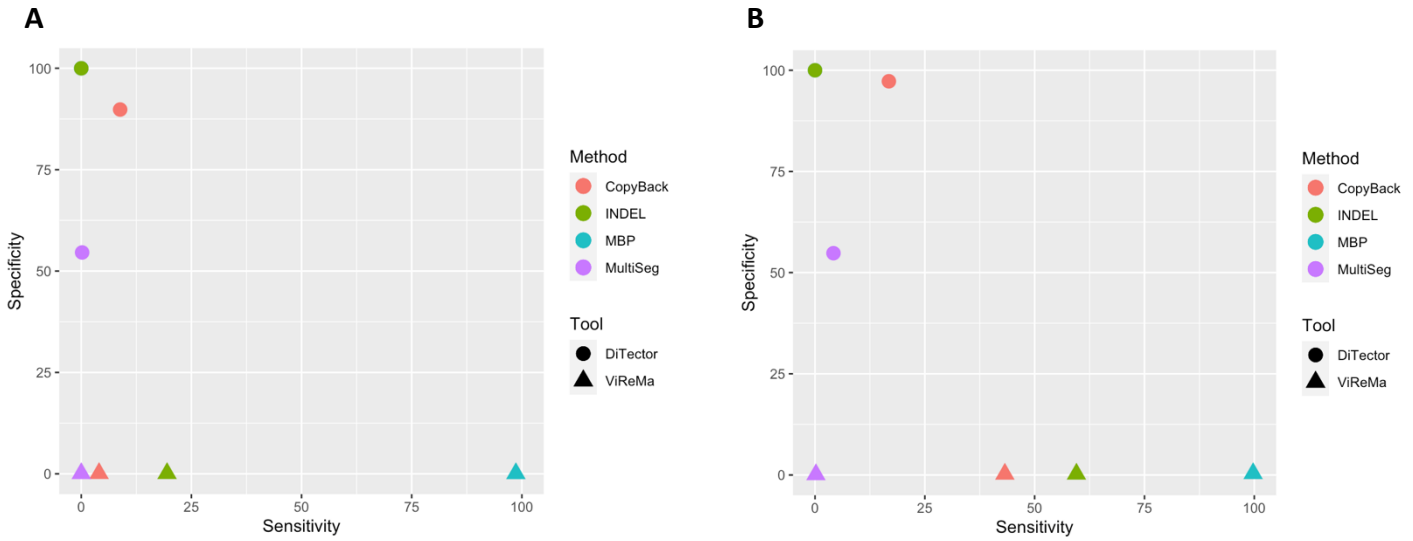


Figure 2: Scatter graph to illustrate sensitivity versus specificity with specificity on the y axis and sensitivity on the x axis. Points are coloured according to the method by which the data was simulated, and shaped according to the tool used A) when STD = 0, B) when STD = 2

	STD	*MBP*	*Indel*	*Copyback*	*MultiSeg2*
DI-Tector	0	50.00	50.00	53.91	50.01
	2	50.10	50.00	58.20	50.36
ViReMa	0	16.10	6.98	1.71	0.07
	2	81.00	42.10	29.80	0.13

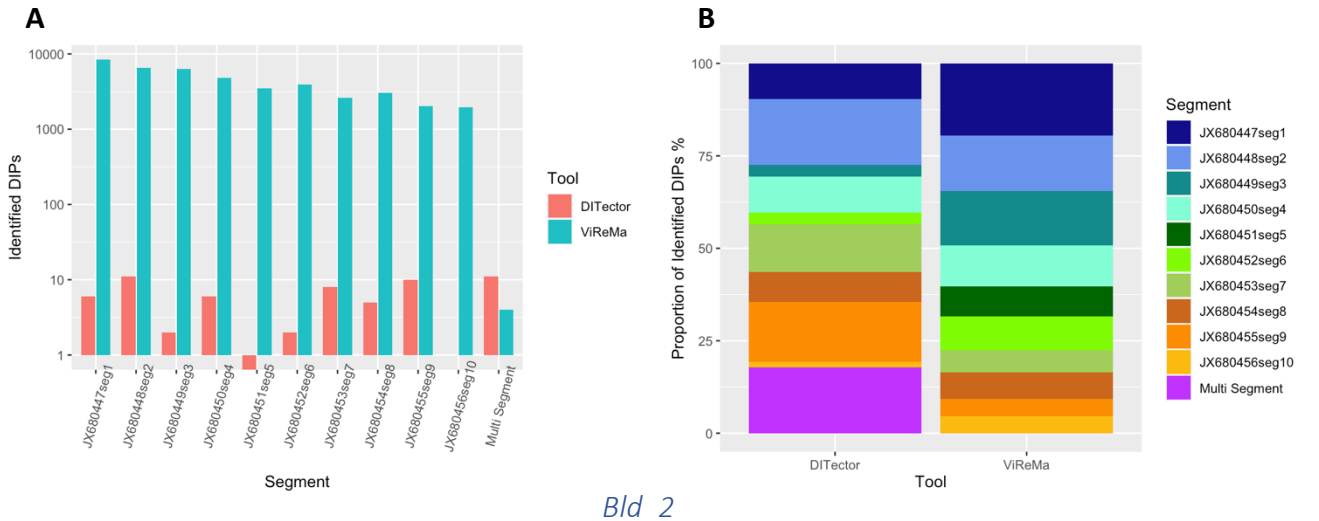
Table 3: Table to show accuracy of tools as a %, in different methods of simulation, with STD = 0, and STD = 2 in Outparse.py.

Tool Comparison on Simulated Data

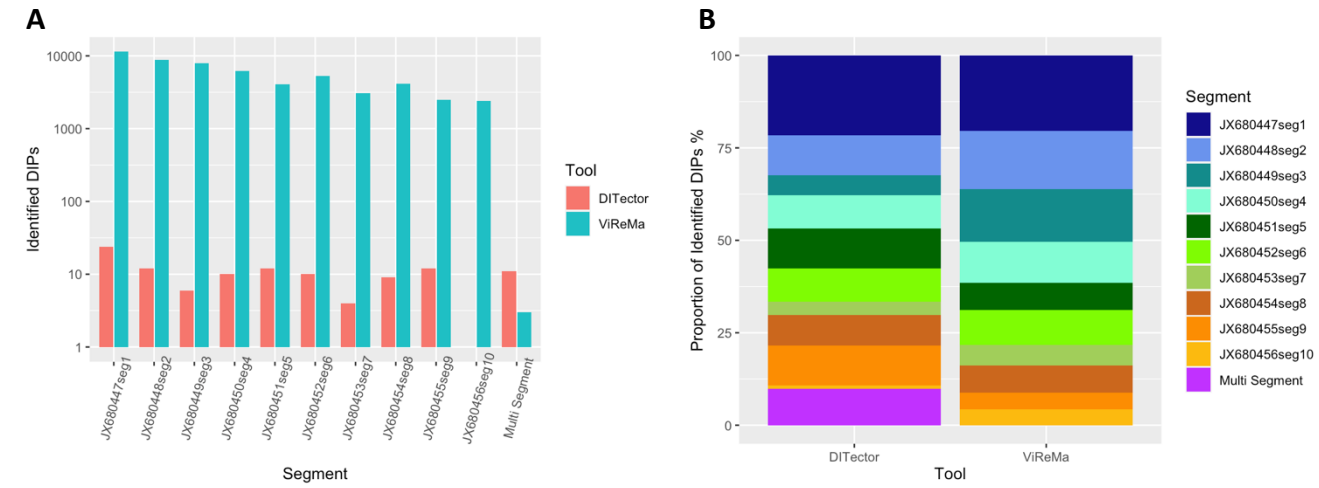
DI-Tector generally found few DIPs but the ones it did identify were correct – false positive rate (<0.01%). ViReMa identified significantly more DIPs but flagged most controls reads as DIPs – false positive rate (99.9%). ViReMa also struggled with the DIPs that spanned across different segments, “MultiSeg2” data where few DIPs were found (Figure 2). Adjusting the STD to 2 did improve the scores on both tools but had varying effects on different DIP types (Figure 2). DI-Tector’s results overall were more accurate (Table 3). ViReMa also had very low accuracy for “Copyback” and “MultiSeg2” methods, arguably the more complex DIP types. Accounting for fuzzing (STD = 2) improved ViReMa’s accuracy significantly across most DIP types, conversely this had little effect to DI-Tector’s results.

DI-Tector took significantly more time to run (on average 711.05 seconds or ~12 minutes) and used significantly more memory, ViReMa took (8.49 seconds).

Bld_1



Bld_2



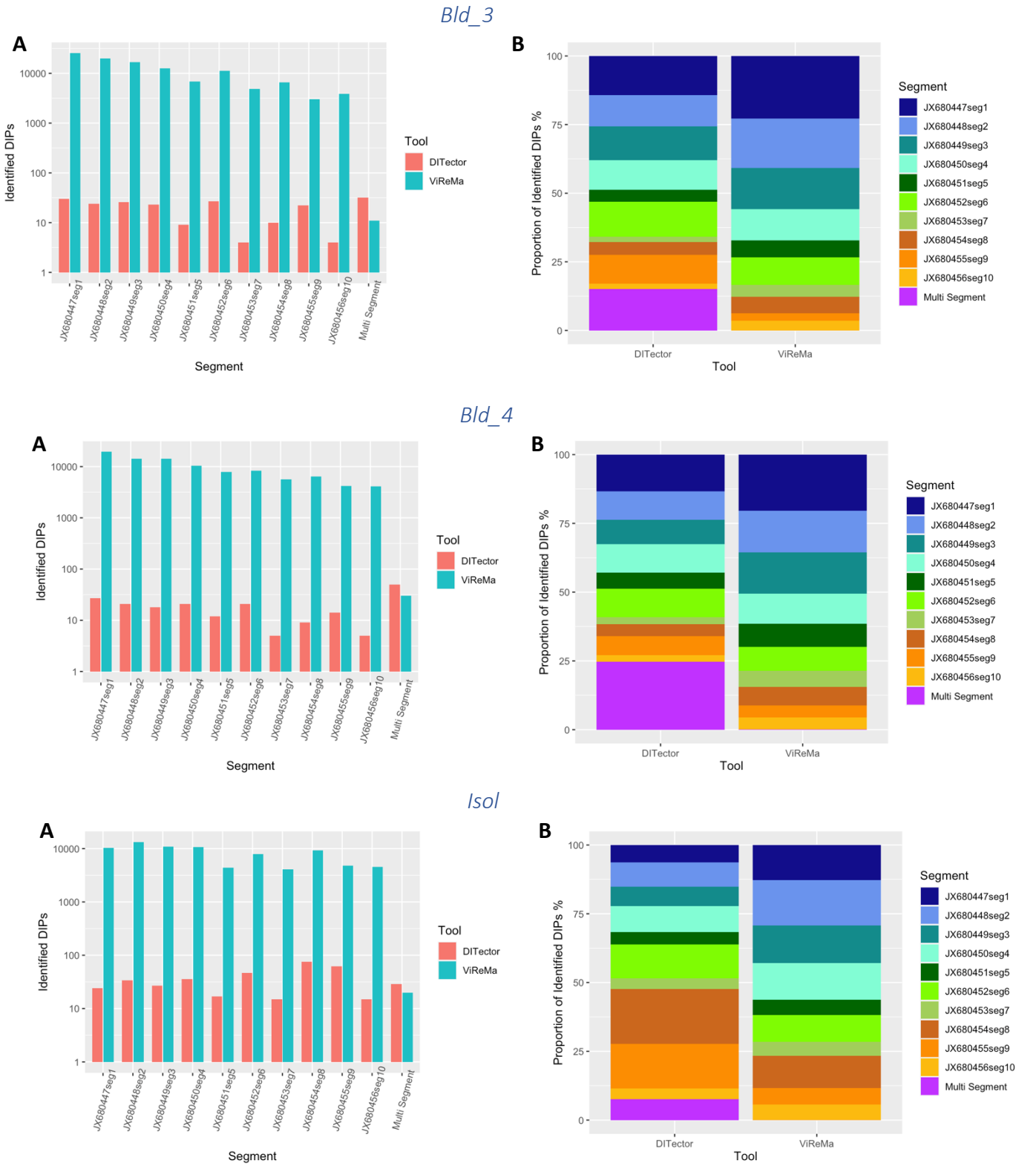


Figure 3: Bld_1, Bld_2, Bld_3 Bld_4, Isol are the samples. A) Bar chart to show the distribution and number of DIPs found between tools and segments, with the number of DIPs identified on the y-axis on a log 10 scale, and the segments across the x-axis. The red bars are reads from DI-Tector, and blue bars from ViReMa. B) Stacked bar chart to show proportion of DIPs found in each segment, with the % of DIPs found in a segment on the y-axis, and the tools on the x-axis, each segment has been coloured differently, and purple notes DIPs that were found across different segments.

2598002

	Bld_1	*Bld_2*	*Bld_3*	*Bld_4*	*Isol*
DI-Tector	62	111	211	203	382
ViReMa	43,081	55,704	111,235	94991	80101

Table 4: Shows number of DIPs found, counting any DIP as 1 regardless of the number of reads that covered a particular DIP

Tool Comparison on Raw Data

Figure 3 A and B shows across all samples there is no trend in proportion of DIPs in a given segment. ViReMa also found significantly more DIPs than DI-Tector (Table 4), however ViReMa found proportionally fewer DIPs across multiple segments (Figure 3.B). This is consistent with the results from the simulated data.

Other outputs from ViReMa found few substitutions, and few insertions. Across all samples the few insertions were 3 bases or less, therefore were accounted as potential sequencing errors rather than DIPs. DI-Tector found mostly insertion type DIPs, and fewer of other types that DI-Tector can detect (Supplementary material: Table 1).

Majority of DIPs were not found in both ViReMa and DI-Tector, only 8 DIPs were found identical on both tools, and were only seen once in each tool.

In Figure 4, most DIPs in the ViReMa bed file are on the negative strand, and the few DI-Tector reads visible are on the positive strand.

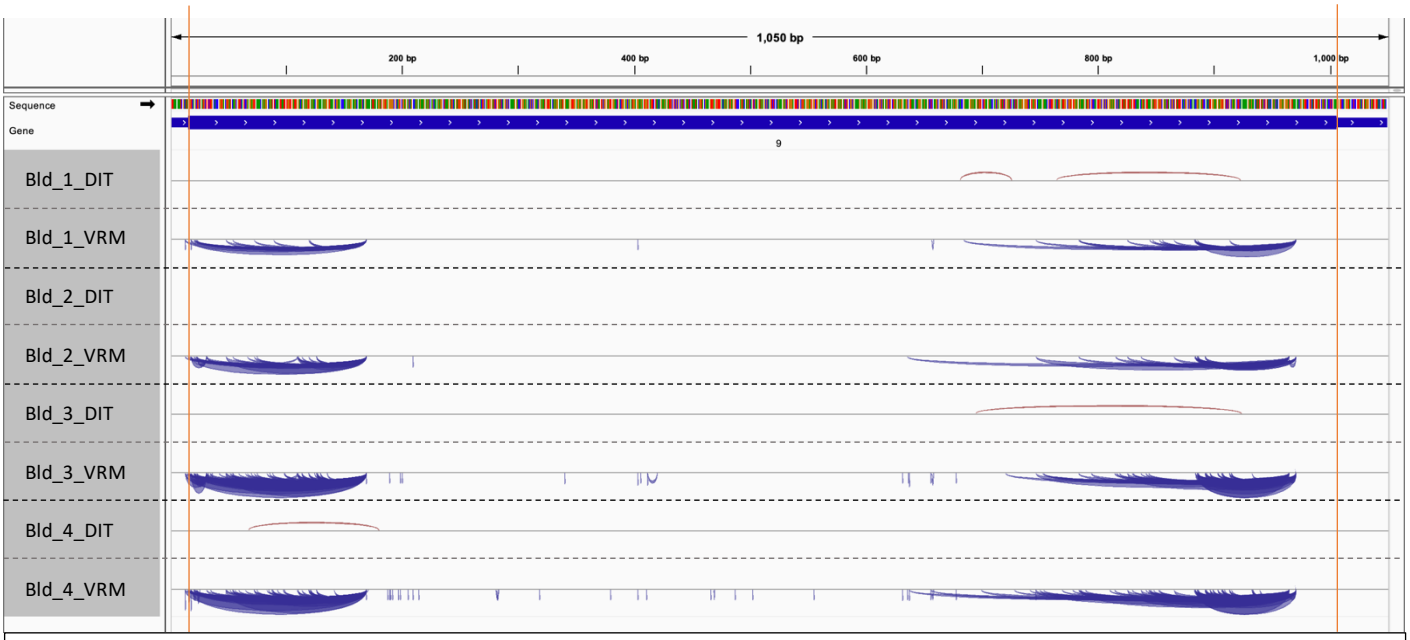


Figure 4: IGV Snapshot of virus recombination events, across segment JX680455seg9, DIT = DI-Tector, VRM = ViReMa, orange lines represent the start and end of the transcription region. All other segments can be found in the Supplementary Material.

BTV Sample Comparison

Table 4 shows that on both tools more DIPs were found in Bld_3 and Bld_4, compared to Bld_1 and Bld_2. Figure 4 shows that between Blood samples there are similar patterns in BrP and RI's, they also predominantly occur in the translation region. The majority of the DIPs either start near the 5' end of the translation region or start at the 3' end of the translation region. Fewer DIPs occurred in the middle portion of the translation region. Deletion size also varied greatly, on single segments BrP and RI had hugely varying distances (Table 5). Samples had on average similar deletion sizes, but samples that produced more DIPs also had DIPs with much larger deletions.

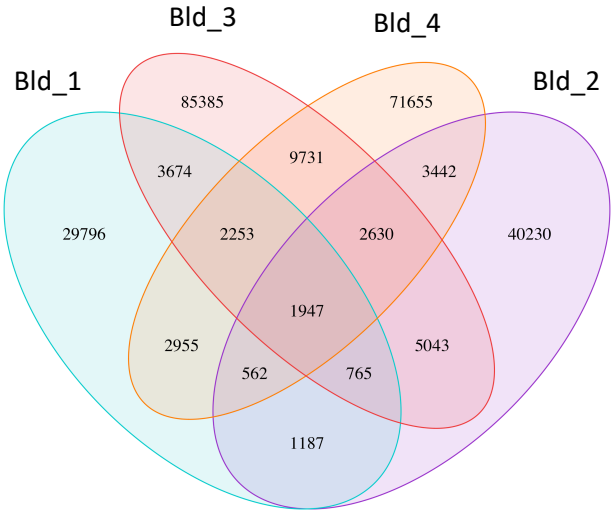


Figure 5: Venn Diagram to compare DIPs between samples, showing combined DIP count found from both tools

Figure 5 shows a large majority of DIPs found were individual to their samples. It also shows that Bld_3 and Bld_4 share the most DIPs, although this is likely because they both have a higher number of DIPs overall.

Figure 6 shows that there are many more segment relationships in Bld_3 and Bld_4 than in other samples. There are not any clear trends in relationships between segments, however there are more relationships seen between segment 1 and other segments, this is likely owing to the fact it is the biggest segment.

When comparing the Isol and Bld_3 sample, Figure 7 shows that the majority of DIPs are different between the samples. Some of these could be attributed to fuzzing, as exact BrP and RI were compared.

	Min	*1st Qu*	*Median*	*Mean*	*3rd Qu*	*Max*
Bld_1	1	33	80	101	148	604
Bld_2	1	39	90	111	161	1235
Bld_3	1	37	87	108	156	1526
Bld_4	1	32	78	100	145	2418

Table 5: Table to show deletion size statistics, Min = minimum, Max = maximum, Qu = quartile

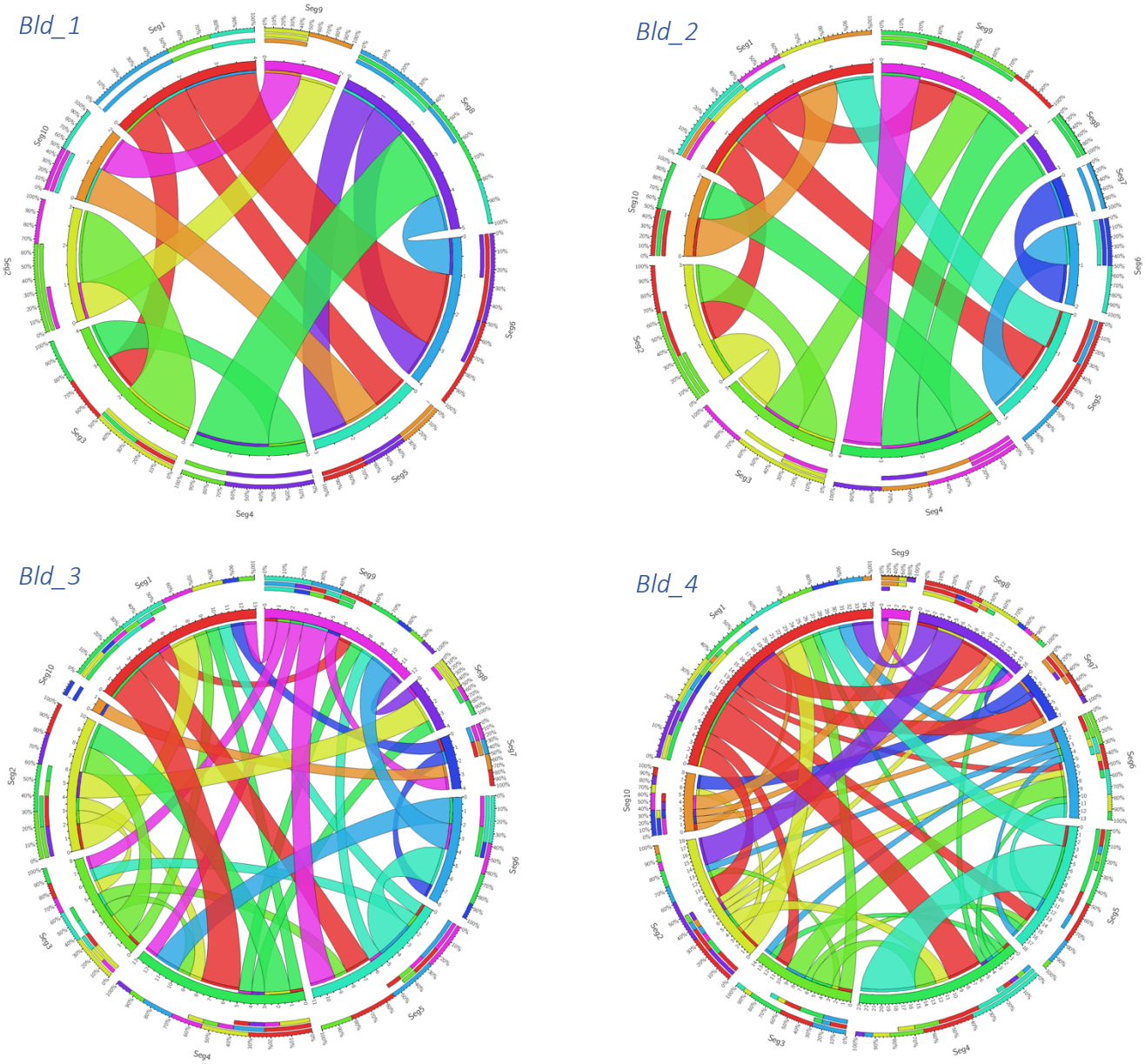


Figure 6: Circos plots to show how DIPs are distributed when across different segments in different blood samples. Each genome segment taking up an area on the circumference. The thicker the line, the higher the percentage of DIPs that take part in that segment-to-segment interaction.

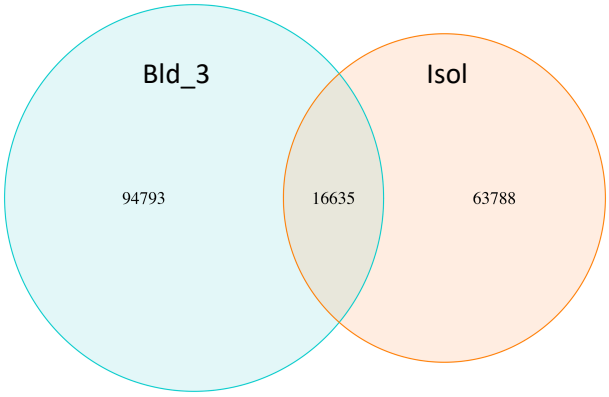


Figure 7: Venn Diagram to compare DIPs found in the Bld_38 and Isol sample, showing the number of combined DIPs found from both tools

Discussion

In this study the available tools, ViReMa and DI-Tector, have been benchmarked for detecting DIPs in NGS data. Two tools were not freely available (Paparazzi and DVG Profiler). For independent benchmarking of the tools from the original supporting papers, a suite of tools was developed for simulating different types of DIPs. DIG.py was developed to compare the outputs from ViReMa and DI-Tector, and establish their ability to identify the simulated DIPs. It can generate deletion, copy-back and snap-back DIPs in 5 different methods: “MBP”, “INDEL”, “Copyback”, “MultiSeg”, “MultiSeg2”, and control WT reads. A set of utility tools were also created which facilitated the comparison of outputs (OutParse.py, CompDIP.py, and ditToBed.py).

DIG.py was effective in generating the simulated reads and could be used to test tools with different viral genomes. Especially being able to track locations of BrP and RI in reads that contain them. Also, looking at DIPs found in the raw BTV data, compared to simulated DIP methods created, showed an accurate representation of many of the DIPs types found in BTV. OutParse.py was useful to compare tool outputs to find convergent DIPs between tools. CompDIP.py was also a useful tool to compare DIPs between samples. Users could use these tools to test performance of tools on different viral genomes. Both OutParse.py and CompDIP.py gave output formats that were much easier for data handling and visualisation in R.

ditToBed.py is a useful tool for any user as it only requires the raw DI-Tector output, allowing easy visualisation of the recombination events in IGV.

Useful bash scripts outlining a pipeline for analysis from raw data to OutParse.py outputs for visualisation, and R scripts for graph outputs can be found on (https://github.com/janewalls/DIG).

The ViReMa paper did not specifically test for specificity but recorded a sensitivity of 59%. This aligned with the results from the “MBP” and “INDEL” methods whilst accounting for fuzzing (STD=2). These methods are similar to the type of simulated data ViReMa was tested on (Routh and Johnson, 2014). DI-Tector was found to have a drastically lower sensitivity than noted in the released paper, but similar specificity results. Sensitivity for copy-back reads 150 nucleotides long: this paper- 8.83%, Original study- ~86%. Though the DI-Tector paper only used 5’ copy-back DIP sequences to test

sensitivity and therefore is not representative of all DIP types (Beauclair et al., 2018). For this reason, independently benchmarking is valuable to directly compare tools with an un-bias approach using more DIP types. Especially as there is limited knowledge as what would be found in BTV. The benchmarking implemented here will provide a more informed decision when choosing a tool for DIP analysis. This study shows that DI-Tector is the better tool to achieve more accurate DIPs and less bias for the type of DIP found. However, to see as many deletion events as possible, ViReMa would be a better tool at the expense of detecting spurious DIPs.

All BTV samples found DIPs, this can provide promising evidence that there are DIPs produced during replication of BTV, which previously had only been theorised (Cowley and Gorman, 1987). No conclusions can be made on the effect of the DIPs on the pathogenicity as none of the cows in the study showed symptoms. Yet previous studies have noted that cows infected with BTV can be asymptomatic (Barratt-Boyes and Maclachlan, 1995). This study also shows that the DIPs primarily occur in the transcription region of the genome, which agrees with findings in previous studies on other viruses such as influenza (Laske et al., 2016).
This study also shows encouraging evidence that DIPs often do not occur at specific BrP and RI sites, given that the majority of DIPs are individual to each sample (Figure 5).

Due to time constraints not all possible features were added to the tools created in DIG. Such as maintaining traceability of BrP and RI in reads whilst doing a subsequent NGS read simulation with base qualities. Also, more detailed information that specific tools provide could be considered by OutParse.py, such as DIP type. Currently CompDIP.py requires the user to have run OutParse.py with both detector tools. An improved functionality would omit this necessity. This would allow a user to use CompDIP.py if they were running one tool, and directly compare samples. ditToBed.py made a BED file modelled on the ViReMa BED file, so making the program create a more detailed BED file would be useful. Additionally, putting the utility tools in the DIG.py package would make installation easier for the user, and they would only have to ensure DIG.py was executable. Finally, better visualisation of results would be to show read alignment with the recombination junctions, for example using a Sashimi plot (Katz et al., 2015). Also, to

be able to view DIPs that occur across segments, to see where in the different parts of segments the BrP and RI are located.

A significant limitation was that the tools were difficult to compare as they did not provide the same specific data. Consequently, factors were not accounted for during analysis to allow for a direct comparison. This could mean invalid comparisons are being made between tools. More analysis could be done on ViReMa using the more recent version and using Bowtie (the default aligner). Specifically, to test if this decreases rate of false positives. Using Bowtie may also decrease run time (Langdon, 2013). More study could also compare the accuracy of DI-Tector's identified DIP type, and the simulated DIP type. Other analysis could also be done to optimise the flags for the individual tools and testing the best performance across different methods, this could then be used to run on raw data. Finally, repeating the simulation multiple times to get more reliable results for sensitivity, specificity, and accuracy, could assist with more robust conclusions on benchmarking a tool.

Further study could investigate other pathogenic effects of disease and DIPs in the cows. For example, the length of time the cows were infected, the WT viral load and whether the cow survived the disease. These factors could give more insight on the possible effect of DIPs on the pathogenicity of Bluetongue disease in the cows.
Many more DIPs were found in Bld_3 compared to KC cells. This indicates the virus produces more DIPs when infecting the host, compared to when in the vector. Which is likely owing to the WT virus replicating more overall once infected in the cow (MacLachlan, 2004). More studies could compare Isol and Bld_3 sample using bed graphs in IGV for comparison of BrP and RI patterns. Further exploration could investigate the deletions that occur, i.e., what specific genes or transcription regions are being deleted. Also, what effect this has on proteins in translation, leading to further understanding of DIP mechanisms.

Additional studies looking at DIPs overall could consider better detection programs to identify DIP reads more accurately from NGS data. A benefit to developing an accurate tool is that there is already an abundance of NGS data available, so more studies could

analyse existing data for DIPs. Also, most studies have looked at influenza, with few studies being done on other viruses (Ziegler and Botten, 2020). Future tools could also look at employing machine learning models to assist in the detection of DIPs, this has already been harnessed in other aspects of NGS technology (Pezoulas et al., 2021).
On the whole, NGS has allowed us to study the DIPs much more effectively and lead the journey to understand the importance and relevance in the effect DIPs have on the WT virus (Alnaji et al., 2018).

Conclusion

DIG provides a useful set of tools to benchmark DIP prediction tools. This also provides guidance on a suitable pipeline for someone considering studying DIPs. Further studies and improvement will be necessary to advance the accuracy of the DIPs BrP and RI sites. This study has shown some novel discoveries, providing evidence for DIPs in BTV. More studies could be done on different viruses and comparing tool outputs. Also, more analysis on BTV could see an effect of DIPs on pathogenicity.

Acknowledgements

Thank you to Dr. Kyriaki Nomikou for processing and providing the raw NGS data, and Dr. Joseph Hughes for his guidance and supervision.

References

Alnaji, F.G., Holmes, J.R., Rendon, G., Vera, J.C., Fields, C., Martin, B.E., and Brooke, C.B. (2018). Illumina-based sequencing framework for accurate detection and mapping of influenza virus defective interfering particle-associated RNAs (Microbiology).

Alnaji, F.G., Holmes, J.R., Rendon, G., Vera, J.C., Fields, C.J., Martin, B.E., and Brooke, C.B. (2019). Sequencing Framework for the Sensitive Detection and Precise Mapping of Defective Interfering Particle-Associated Deletions across Influenza A and B Viruses. J Virol *93*.

Andrews, S. (2010). Babraham bioinformatics-FastQC a quality control tool for high throughput sequence data. URL: Https://Www. Bioinformatics. Babraham. Ac. Uk/Projects/Fastqc.

Aoki, H., Ishikawa, K., Sekiguchi, H., Suzuki, S., and Fukusho, A. (2003). Pathogenicity and kinetics of virus propagation in swine infected with the cytopathogenic classical swine fever virus containing defective interfering particles. Archives of Virology *148*, 297–310.

Barratt-Boyes, S.M., and Maclachlan, N.J. (1995). Pathogenesis of bluetongue virus infection of cattle. Journal of the American Veterinary Medical Association (USA).

Beauclair, G., Mura, M., Combredet, C., Tangy, F., Jouvenet, N., and Komarova, A.V. (2018). *DI-tector* : defective interfering viral genomes' detector for next-generation sequencing data. RNA *24*, 1285–1296.

Bosma, T.J., Karagiannis, K., Santana-Quintero, L., Ilyushina, N., Zagorodnyaya, T., Petrovskaya, S., Laassri, M., Donnelly, R.P., Rubin, S., Simonyan, V., et al. (2019). Identification and quantification of defective virus genomes in high throughput sequencing data using DVG-profiler, a novel post-sequence alignment processing algorithm. PLoS ONE *14*, e0216944.

Cock, P.J., Antao, T., Chang, J.T., Chapman, B.A., Cox, C.J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., et al. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics *25*, 1422–1423.

Cowley, J.A., and Gorman, B.M. (1987). Genetic reassortants for identification of the genome segment coding for the bluetongue virus hemagglutinin. J Virol *61*, 2304–2306.

Dimmock, N.J., and Easton, A.J. (2014). Defective Interfering Influenza Virus RNAs: Time To Reevaluate Their Clinical Potential as Broad-Spectrum Antivirals? Journal of Virology *88*, 5217–5227.

Huang, A.S., and Baltimore, D. (1970). Defective Viral Particles and Viral Disease Processes. Nature *226*, 325–327.

Jennings, P.A., Finch, J.T., Winter, G., and Robertson, J.S. (1983). Does the higher order structure of the influenza virus ribonucleoprotein guide sequence rearrangements in influenza viral RNA? Cell *34*, 619–627.

Johnson, D.M., Cubitt, B., Pfeffer, T.L., de la Torre, J.C., and Lukashevich, I.S. (2021). Lassa Virus Vaccine Candidate ML29 Generates Truncated Viral RNAs Which Contribute to Interfering Activity and Attenuation. Viruses *13*, 214.

Katz, Y., Wang, E.T., Silterra, J., Schwartz, S., Wong, B., Thorvaldsdóttir, H., Robinson, J.T., Mesirov, J.P., Airoldi, E.M., and Burge, C.B. (2015). Quantitative visualization of alternative exon expression from RNA-seq data. Bioinformatics *31*, 2400–2402.

Krueger, F. (2015). Trim galore. A Wrapper Tool around Cutadapt and FastQC to Consistently Apply Quality and Adapter Trimming to FastQ Files *516*, 517.

Krzywinski, M., Schein, J., Birol, I., Connors, J., Gascoyne, R., Horsman, D., Jones, S.J., and Marra, M.A. (2009). Circos: An information aesthetic for comparative genomics. Genome Research *19*, 1639–1645.

Langdon, W.B. (2013). Which is faster: bowtie2 ^{GP} bowtie > bowtie2 > BWA. In Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, (Amsterdam The Netherlands: ACM), pp. 1741–1742.

Langmead, B., and Salzberg, S.L. (2012). Fast gapped-read alignment with Bowtie 2. Nature Methods *9*, 357–359.

Laske, T., Heldt, F.S., Hoffmann, H., Frensing, T., and Reichl, U. (2016). Reprint of "Modeling the intracellular replication of influenza A virus in the presence of defective interfering RNAs. Virus Research *218*, 86–95.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). The sequence alignment/map format and SAMtools. Bioinformatics *25*, 2078–2079.

MacLachlan, N. (2004). Bluetongue: pathogenesis and duration of viraemia. Vet Ital *40*, 462–467.

Maclachlan, N.J. (2011). Bluetongue: History, global epidemiology, and pathogenesis. Preventive Veterinary Medicine *102*, 107–111.

von Magnus, P. (1954). Incomplete Forms of Influenza Virus. In Advances in Virus Research, (Elsevier), pp. 59–79.

Manzoni, T.B., and López, C.B. (2018). Defective (interfering) viral genomes re-explored: impact on antiviral immunity and virus persistence. Future Virology *13*, 493–503.

Marriott, A.C., and Dimmock, N.J. (2010). Defective interfering viruses and their potential as antiviral agents: Defective interfering viruses. Rev. Med. Virol. *20*, 51–62.

Mercado-López, X., Cotter, C.R., Kim, W., Sun, Y., Muñoz, L., Tapia, K., and López, C.B. (2013). Highly immunostimulatory RNA derived from a Sendai virus defective viral genome. Vaccine *31*, 5713–5721.

Pezoulas, V.C., Hazapis, O., Lagopati, N., Exarchos, T.P., Goules, A.V., Tzioufas, A.G., Fotiadis, D.I., Stratis, I.G., Yannacopoulos, A.N., and Gorgoulis, V.G. (2021). Machine Learning Approaches on High Throughput NGS Data to Unveil Mechanisms of Function in Biology and Disease. Cancer Genomics Proteomics *18*, 605–626.

Routh, A., and Johnson, J.E. (2014). Discovery of functional genomic motifs in viruses with ViReMa–a Virus Recombination Mapper–for analysis of next-generation sequencing data. Nucleic Acids Research *42*, e11–e11.

Thorvaldsdottir, H., Robinson, J.T., and Mesirov, J.P. (2013). Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. Briefings in Bioinformatics *14*, 178–192.

Vasilijevic, J., Zamarreño, N., Oliveros, J.C., Rodriguez-Frandsen, A., Gómez, G., Rodriguez, G., Pérez-Ruiz, M., Rey, S., Barba, I., Pozo, F., et al. (2017). Reduced accumulation of defective viral genomes contributes to severe outcome in influenza virus infected patients. PLoS Pathog *13*, e1006650.

Vignuzzi, M., and López, C.B. (2019). Defective viral genomes are key drivers of the virus–host interaction. Nat Microbiol *4*, 1075–1087.

Vodovar, N., Goic, B., Blanc, H., and Saleh, M.-C. (2011). *In Silico* Reconstruction of Viral Genomes from Small RNAs Improves Virus-Derived Small Interfering RNA Profiling. J Virol *85*, 11016–11021.

Wei Li (2015). getinsertsize.py. https://gist.github.com/davidliwei/2323462#file-%20getinsertsize-py

Ziegler, C.M., and Botten, J.W. (2020). Defective Interfering Particles of Negative-Strand RNA Viruses. Trends in Microbiology *28*, 554–565.