Aneuploid chromosome copy number estimation with dynamic coverage analysis

Steps towards improved short read aneuploid assembly

Manuel Carbajo Martínez

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday April 12th, 2017 at 13:30 AM.
Leiden student number: s1601504
Delft student number: 4429427
Project duration: March 1, 2012 – January 1, 2013
Thesis committee: Prof. Marcel Reinders, TU Delft, Head of the Pattern Recognition & Bioinformatics Group
Assistant Prof. Thomas Abeel, TU Delft Supervisor
Dr Erwin Bakker, LIACS Supervisor
Tamim Abdelaal, Phd Student

An electronic version of this thesis is available at
https://github.com/AbeelLab/Pedca/tree/master/Documents

Abstract

Motivation: With the decreasing cost and increasing read length of next-generation sequencing technologies, high quality draft genome sequences are being published at an increasingly faster rate. With well-established methodologies, larger and more complex genomes are being tackled. However, the ploidy of the genome (the copy number of its chromosomes) greatly affects the accuracy of the assembly. While genomes with only one (haploid) or two (diploid) copies of each chromosome have existing efficient methodologies to assemble them, polyploid (constant ploidy higher than two) and aneuploid (variable copy number per chromosome) haplotyping is much more challenging to solve. Given the similarity between multiple copies of a basic genome in polyploid individuals, the assembly of such data usually results in collapsed contigs that represent a variable number of homologous genomic regions. Such collapse is far from ideal, as the corresponding lost information leads to inaccurate assemblies and a corresponding lack of precision in understanding how haplotypes influence phenotype. Here we address the question of how to determine the precise ploidy of a genome, which is the first step for solving a fully detailed aneuploid assembly. We do so by integrating a range of current and new features to improve an already existing approach: depth of coverage analysis. We also briefly address some probable leads to solve polyploid haplotyping.

Results: We introduce PEDCA (Ploidy Estimation by Dynamic Coverage Analysis), an algorithm to improve the quality of short reads aneuploid genomes assembly by estimating the copy number of their different contigs. PEDCA infers the ploidy of the fragments by analyzing the coverage distribution of the aligned reads and the ratio among its clusters. It simplifies previous methods which improves its exportability to a wide spectrum of genomes with different characteristics, and adds a set of extra features to improve the ploidy analysis. As a proof of concept, PEDCA is tested on simulated as well as real sequencing data from CBS1483 (*S.pastorianus* yeast). Finally, we apply our method to infer the chromosome copy number of *Trypanosoma cruzi* CL Brener for which the copy number is yet unknown. We also briefly explore some ideas for a draft complementary method of polyploid haplotyping. This second algorithm, while not completed, examines and offers new paths to avoid a very common problem in polyploid phasing: the exponential computation of all possible haplotype combinations.

Availability: PEDCA can be downloaded at: https://github.com/AbeelLab/Pedca

Contact: t.abeel@tudelft.nl

Table of Contents

Presentation

Diploid organisms, for example humans, have two copies of each chromosome. While most of the genetic sequence from each haplotype is the same, differences among each copy appear at specific loci. Other organisms, like many prokaryotes and certain animals and plants (among them many important crop plants like wheat, potatoes, oat or soy), are *polyploid* ("The Polyploidy Portal," n.d.). These describe a constant number of chromosomes (but higher than two) for the whole genetic content. Following the same logic, an *aneuploid* genome has an irregular copy number for each of its chromosomes, with different levels of heterozygosity. The genomes that this research tries to improve are aneuploid but the method can also be applied to polyploid organisms.

The differences among the alternative haplotypes might range from one single nucleotides (SNP's, which are the most frequent type, including deletions or inserts), to very large structural variations (duplications, translocations, inversions…). *Genotype calling* reports the variant sites without associating them to a specific allele, while *haplotyping* or *phasing* provides the order of the variations in each allele. This extra information is crucial for understanding all of the potential genetic expressions of the haplotype which affect the metabolism of the organism. Fundamental applications rely on this description, from determining the appearance or absence of physiological traits, to identifying different susceptibility to diseases or treatments, or understanding recombination patterns of genetic inheritance (Wagner, n.d.).

Finding methods that specifically target the detailed sequence analysis of aneuploid genomes is a central challenge to understanding these organisms. Many types of yeast, which are used for a large number of biological applications like fermentation or bio-fuel and biopolymers production, have aneuploid genomes.

To sequence a given genome, multiple copies of its DNA are segmented into a large amount of reads. Current algorithms reconstruct the original genome in a single sequence based on the consensus of the most represented variations in the overlapping reads.

In this thesis, we assume that at least a draft *de novo* assembly exists or can be built from the available reads. Therefore, we split the problem in to two complementary questions that need to be answered in order to solve the detailed sequence of each haplotype. The first one determines the number of copies that each chromosome or *contig* has, which is also known as its *ploidy*. The second one finds the variations among each of the copies and how they cluster within each of the haplotype, which is also referred to as *phasing* or *haplotyping*. We focus mainly on the first question. We fully address ploidy estimation and provide a new algorithm that improves previously existing approaches. We test it in simulated and real datasets, some of which could not be solved with previously existing methods. We also briefly address the second one (aneuploid phasing), exploring a possible solution that avoids a common problem found in polyploid haplotyping, but our line of thought doesn't successfully conclude in an operational software. Nevertheless, we found it important to describe the main ideas that were followed and to report the difficulties that we encountered when trying to implement them.

An introduction to ploidy estimation

While parallel regions amongst different copies of a particular chromosome are by definition very similar, each copy is likely to have its own specific variations. Averaging intra-chromosomal variations therefore results in a flattened homomorphic genome with its consequent loss of information.

Aneuploid organisms have a varying chromosome copy number; therefore, in order to describe in detail the sequence of each haplotype, we first need to infer the correct number of copies that each chromosome has. Copy number variation is often referred to in the literature as the number of times that a certain genetic region is repeated within a genome reference. In this sense, copy number variation is a concept referring to a structural variation. But in this report, we refer to *chromosomal copy number variation* or *ploidy* as the number of copies that each contig or chromosome has. We infer the *chromosome copy number* by analyzing the *coverage*, which is the number of times that a very similar chromosome or *contig* sequence is supported by the reads along the genome. In order to infer the number of times that each contig is repeated, we need to know its sequence. We use the *consensus assembly* sequence as a reference.

A *consensus assembly* describes a fair approximation of a genome where only the most frequent variations are represented. When a new assembly is performed, the small differences corresponding to the specific individual genetic signature of the specimen will arise, but when reconstructing the genome, the small reads have to be merged into longer contiguous sequences called *contigs*. The general approach is to flatten out differences by consensus. The most represented variations will determine the consensus assembly with the less represented variations being flattened and disappearing. We use the term *consensus reference* for the sequences obtained using this method.

When no *reference genome* is available, a *draft de novo reference* can be built with a *de novo* assembler. There are a number of different *de novo* assemblers available and they use two main approaches to merge the reads into longer contigs sequences: *String graphs Overlap-Layout-Consensus* (Myers, 1995) (Myers, 2005) and *de Bruijn* graphs (Pevzner, Tang, & Waterman, 2001).

(Alkan, Sajjadian, & Eichler, 2011) underline two general problems that limit the quality of the assemblers: contamination from other organisms' DNA and repeat content. Contamination rates are difficult to estimate, especially in *de novo* assemblies; given the irreversible tendency for cheaper and faster technology to sequence genomes, the importance of these errors is most likely increasing. But it is very difficult to address or even measure the extent of the contamination problem when no reference is available. We'll be encountering much more clearly the problem of repeated regions in this research. Using mate pairs reads to resolve repeats has been shown to be effective (Wetzel, Kingsford, & Pop, 2011), but very high repeat content in genomes still poses a problem in *de novo* assemblies, especially when very long regions are involved. This particular aspect can affect our method for solving both chromosome copy numbers (since our approach is based on coverage analysis) and the phasing problem (since the true position of the different variations might be hard to define).

The Assemblathon papers (Earl et al., 2011) (Bradnam et al., 2013) describe a set of different evaluation methods to assess the performance of assemblers. While the purpose of this paper is not to reevaluate different assembly methods, we found it important to become familiarized with some commonly used software, to understand how they perform and affect the data that we would be using

later. We used the Newbler software package (454 Life Sciences, Branford, CT), for assembling Illumina short reads libraries, but there was a large range of choice for this step. We also performed assemblies with ALLPATHS and SOAPdenovo and used measures such as the N50 and the number of misassemblies to evaluate their results.

We can then start to address the problem of copy number estimation. Correct chromosomal copy number knowledge is a prerequisite to understanding a given genome. Knowing the ploidy of a sequence helps us to search and identify the number of potential variations present in a segment. Those variations might be essential to understanding the phenotypes of the organism.

There are basically two algorithmic approaches to solve the chromosome copy number problem: The stochastic model and depth of coverage analysis.

The stochastic approach analyzes variations among reads. It estimates which observations can be considered errors and which ones are more likely to be an alternative version of the same chromosome; this allows us to infer their copy number.

The depth of coverage analysis infers the number of times a contig occurs in the genome from sequenced read depths. The intuition is quite straightforward: assuming that the short reads are randomly sampled on the genome, a chromosome will have a number of reads that map to its flattened consensus reference proportional to its copy number. The changes in the coverage over the genome follow a proportional ratio that allows the inference of the different ploidies (**Figure 1**).

We propose PEDCA, an improved depth of coverage method, to allow detailed copy number estimation per contig in *de novo* aneuploid assemblies. Our method presents itself as a .jar file that works on any operating system with a Java platform. It only requires as an input an alignment file (.bam or .sam) of short reads mapped to a draft reference assembly. An optional complementary analysis can be obtained if a variant call file (.vcf) is provided.

We demonstrate PEDCA's performance by inferring the ploidy of a control set of simulated chromosomes and by applying it to the genome of the yeast *S.pastorianus* strain CBS 1483. We use two sequencing versions of CBS1483. The first is the same that has already been studied and for which we already have some results to compare against ours. The second version could not be solved with previous methods and we use it to illustrate the utility of our implementation. We also apply PEDCA on the genome of *Trypanosoma cruzi* CL Brener, a parasite which causes an infectious disease and which so far has unknown ploidy. This organism has a high repetitive genetic content that provides a very noisy coverage data set which is very useful for testing the limits of our method.

A PEDCA tutorial with illustrated examples can be found at the end of this document (supplemental material **56**) and downloaded at: https://github.com/AbeelLab/Pedca/tree/master/Documents

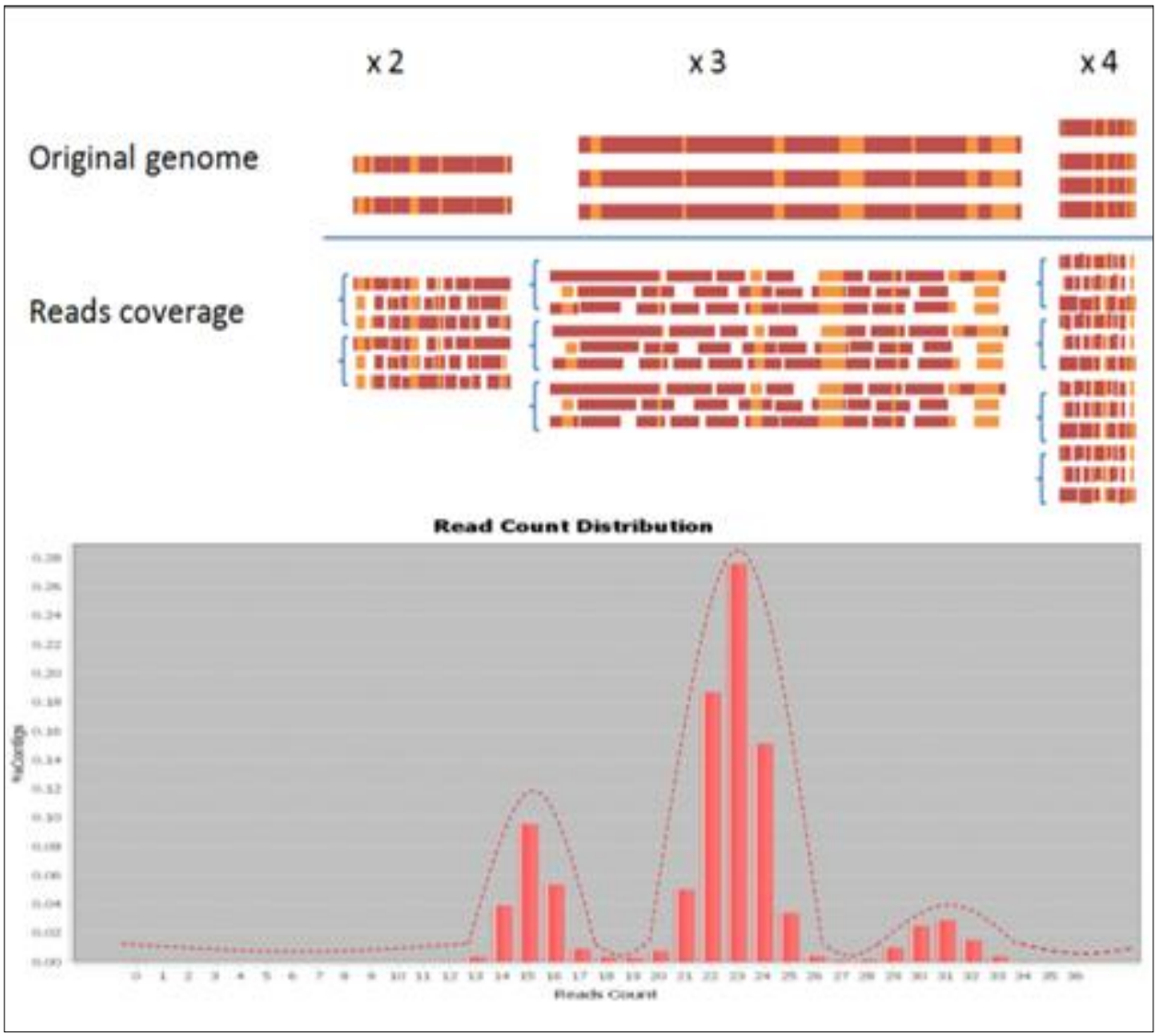


Figure 1 The intuition behind ploidy estimation by coverage analysis. Regions with similar depth have similar read count (top figure). An increase in copy number corresponds to a proportional increment in the number of reads that map to the flattened reference of that region. Reads having the same copy number cluster together around the same normal distribution. Reasoning in the opposite direction, the ratio between the centers of the clusters in the read count distribution allows the inference of the original copy number ratios and by extension their estimation (bottom figure).

Methods: Ploidy estimation

Simulated DNA sequence data. Building a realistic simulated S.*pastorianus* genome

We know that S.*pastorianus* is a hybrid of two ancestral parent yeasts, *S.cerevisiae* and *S.eubayanus* (Libkind et al., 2011). In order to validate our model, we build a simulated aneuploid dataset of five different chromosomes with different copy numbers each. Because we want our simulated data set to serve as a control genome that we can use to validate our method, we build it to be as similar as possible to a real aneuploid genome. While we do not have a full reference assembly for S.*pastorianus*, we do have a fully sequenced genome for one of its ancestor, the *S.cerevisiae* that we can therefore use as a control reference.

We select 5 chromosomes (I, II, III, IV and X) of *cerevisiae*, all with known copy numbers in the *pastorianus* genome (van den Broek et al., 2015). All simulated chromosomes belong to the S288C strain. Out of the five chromosomes, three have a copy number of 3 (Chromosomes I, II and IV) one chromosome has 2 copies (Chromosome X) and the last one has 4 (Chromosome III). Chromosome X has also a 10 Kb deletion which will allow us to evaluate our precision in detecting small structural variations.

Since the copied reference is flattened, all chromosome copies have the same sequence. But we want each different chromosome copy to have its own set of specific variations. In order to introduce realistic alternative alleles, we extract the ones that are present in the real sequenced Illumina reads (four libraries of different insert sizes) that were used to build the reference assembly CBS1483. Before describing how we extract the variations from the real reads and reintroduce them into the flattened copies, let us describe the libraries.

The real libraries are the same ones that were used by (van den Broek et al., 2015) to assemble the CBS1483 reference. They were obtained by sequencing a CBS1483 genome using an Illumina HiSeq2000 sequencer (Illumina, San Diego, CA) at Baseclear (Leiden, The Netherlands). A total of four paired libraries with different insert sizes were obtained. Two 100-cycle paired-end libraries with insert sizes of 500 and 180 bp and two 50-cycle mate pair libraries with 3- and 8-kb insert sizes were sequenced. The combined libraries comprised more than 10^8 reads, which represent approximately 7 Gb, resulting in a genome coverage of +/-270 times (see **Supplemental Table 1**).

Using bwa-0.7.13, we mapped the four libraries of *pastorianus* Illumina reads to the *cerevisiae* reference. The resulting .bam files represent only the genetic heritage of the hybrid genome that maps to the corresponding *cerevisiae* chromosomes (**Figure 2**).

The mapping accepts a certain degree of mismatches per alignment, which is interesting to us because it allows us to align those reads with small variations to regions that have lost their alternative alleles in the consensus assembly. Whenever a loci has different sequence per haplotype, the assembler disregards the minorities and retains the most represented one. We recover the diversity present in the original reads by performing variant calling with Pilon (Walker et al., 2014), mapping the *pastorianus* reads to the *cerevisiae* reference (see statistics on **Supplemental Table 2**). All of the variations (about 3% of the genome's positions) are reported in their corresponding .vcf files. Plotting the distribution of the allele frequencies stored in the .vcf file confirms Magnolya's copy number prediction (**Supplemental Figure 4**).

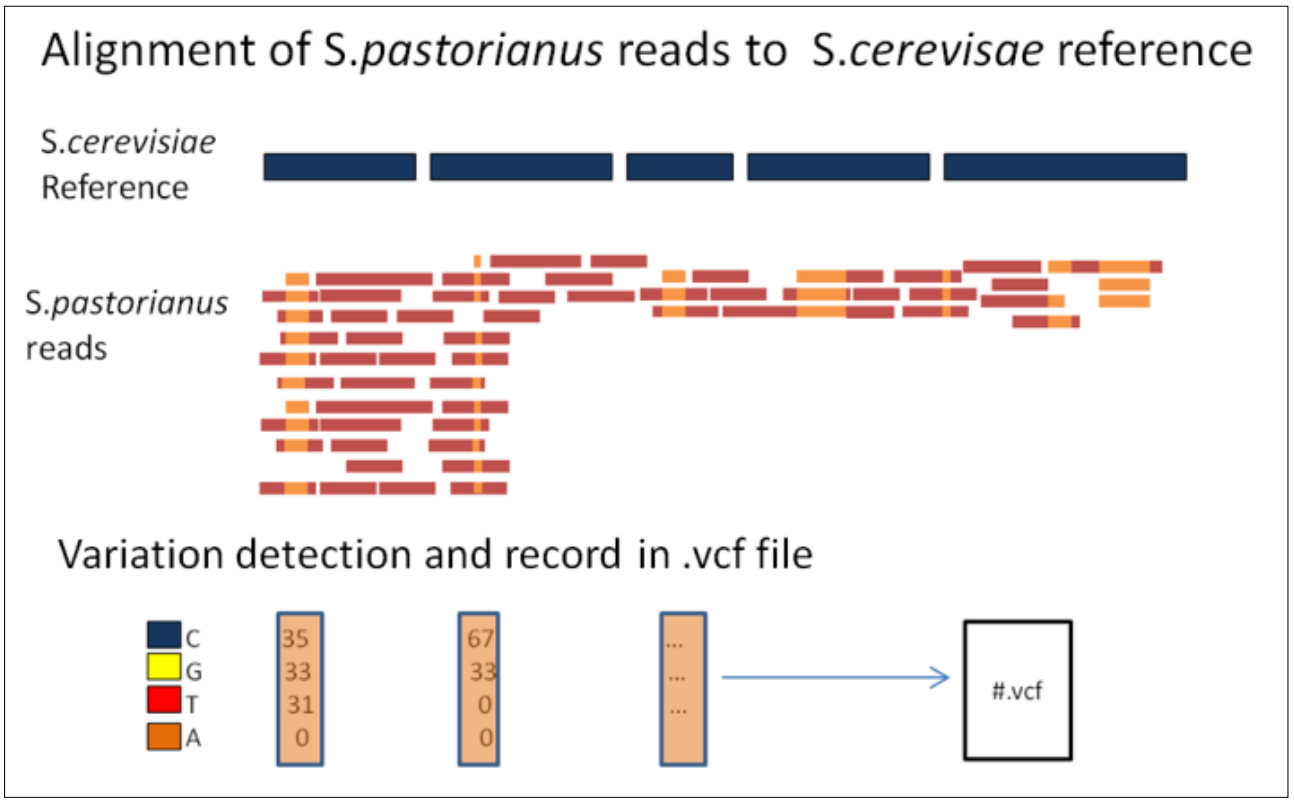


Figure 2 The hybrid nature of S.pastorianus allows us to reconstruct a reliable simulated genome based on the ancestral genetic heritage of the cerevisiae chromosomes. Aligning *S.pastorianus* reads to a *S.cerevisiae* reference allows us to obtain the real variations currently present in the real *pastorianus* genome. The result of the base calling reflects the alternative alleles at the heterozygous positions of the haplotypes which are stored in the corresponding .vcf file, to be used later in the construction of the simulated data.

We introduce the detected variations at the precise loci of the reference where they were reported by Pilon. Because we don't know the correct haplotype to which each variation belongs, they are randomly distributed among all of the possible copies of each chromosome, but always at the exact position where the .vcf file reports them (**Figure 3**). This results in different sequences for each chromosome copy. The copies have the same exact variations as the one detected by Pilon in the real *pastorianus* genome, and at the same exact loci. If the copies are flattened by consensus, they result in the same sequence as the real S288C reference. The only significant difference between our simulated and the real *pastorianus* data is that the variations are differently phased. We can therefore compare the results of our method to a sample control set that is identical in all practical aspects to a significant portion of the real data set that we will be trying to solve.

Once the sequence of the different copies of each simulated chromosome was defined, we simulated sequenced reads from them. The goal was to test our method on these simulated reads to see if we could predict the ploidy of our simulated genome. Following (Escalona, Rocha, & Posada, 2016) read simulators comparison and evaluation, we chose to generate our simulated reads with ART (Huang, Li, Myers, & Marth, 2012), an open source software that can generate synthetic Illumina paired end reads with substitution errors and built-in quality score profiles measured by previous studies. We ran ART with the HS20 platform profile that is provided with the simulator for paired-end reads. ART requires three parameters to generate paired-end reads: the coverage of each haplotype, the mean insert length, and the standard deviation of insert length. We measured these

parameters in our CBS1483 libraries and recreated the simulated reads with similar values (for details on how we obtained these parameters see **Supplemental Figure 1**).

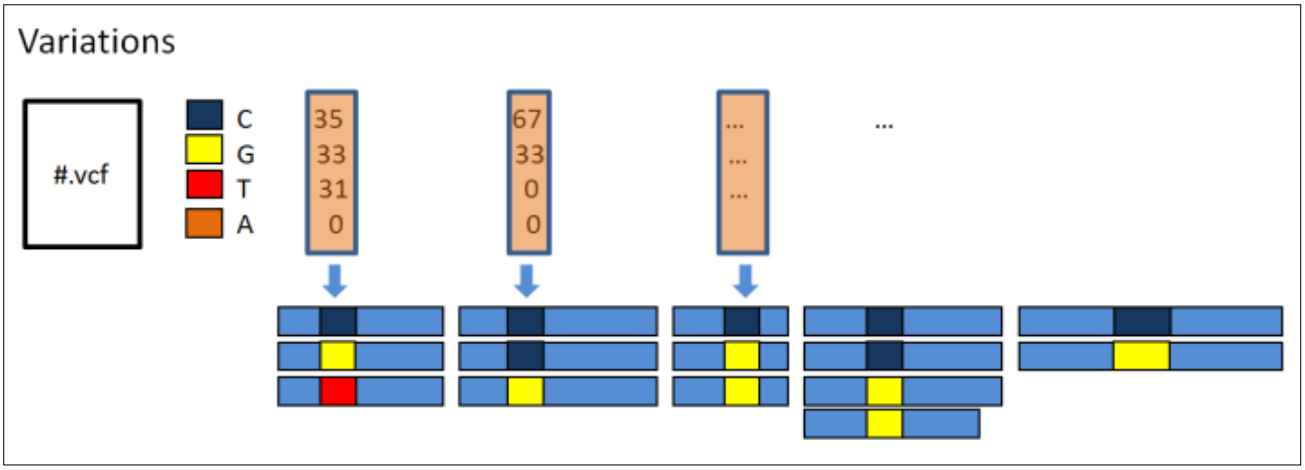


Figure 3 Introducing measured variations into simulated alleles. For every variation position detected, the same proportion of bases is reintroduced into our simulated haplotypes. Before, we only had an identical sequence repeated a certain number of times for each contig. After reintroducing the variations stored in the .vcf file, we have haplotypes that are different from each other. For example, the first position reports a variation of approximately 1/3 of C's, 1/3 of G's and 1/3 of T's, each of which are randomly assigned to any of the three copies of that contig. The second position reports a different ratio of approximately 2/3 C's and 1/3 of G's. Our final simulation contains the same degree of allele frequencies as the original, and at the same exact loci, but distributed randomly among the possible haplotypes.

Real DNA sequence data

We use two other datasets to test our method, both of the *S.pastorianus* strain CB1483. The first is the same one that we had used as a base to build the simulated dataset described above. It consists of four libraries of sequenced Illumina paired end reads with different insert sizes from CBS1483. We'll refer to this sequencing as the BaseClear CBS1483.

We also have a version of the CBS1483 strain sequenced by the company Novogene that was previously unused because it did not work well with Magnolya (van den Broek et al., 2015). We'll use this data set to illustrate the improvements provided by our method. We'll refer to this reference as the Novogene CBS1483.

Finally, we'll also describe the results of PEDCA applied to Illumina reads of *Trypanosoma cruzi* CL Brener, a genome for which an advanced assembly does not exist and where ploidy is unknown.

Consensus draft reference genome

In order to analyze the ploidy of a contig, we need an initial sequence that we can use as a reference. That initial assembly is the result of inferring the right order of a set of sequenced reads by aligning it to a previously existing reference or, if none is available, by performing a *de novo assembly*. There is a wide offer of different *de novo assemblers* available and they use one of the two main approaches to merge the reads into longer contigs sequences: String graphs Overlap-Layout-Consensus (Myers, 1995) (Myers, 2005) and de Bruijn graphs (Pevzner et al., 2001).

We assembled our simulated dataset using three different *de novo* assemblers: Newbler, SOAPdenovo and ALLPATHS. Newbler is a standard 454 Illumina assembler which is based on the *overlap-layout-consensus* algorithm. ALLPATHS is specially designed to assemble Illumina GA short reads and is based on a *de Bruijn* graph approach (Butler et al., 2008). According to its authors, ALLPATHS is suitable for large genomes up to human size. SOAPdenovo (R. Li et al., n.d.) also works using a *de Bruijn* graph using pre-set thresholds for k-mer frequencies.

The PEDCA algorithm: an enhanced depth of coverage algorithm.

PEDCA is built on the same principle as Magnolya, but with a few significant modifications. PEDCA presents itself as an executable .jar file which allows it to run on any operating system with a Java platform. It requires only one input: the bam/sam files of the contigs aligned to the draft reference genome. For our simulation and tests, we mapped our simulated reads against the simulated reference genome used to generate them, obtaining the corresponding .bam file.

Because the coverage analysis is based on a very straightforward logic, we want to find a way to improve its reliability and allow the method to work for a large spectrum of different genomes. We implement different features to do so, such as:

- removing extreme values,
- adding a sliding window to sample depth of coverage over the genome,
- a simplified naïve smoother to fit the read count distribution and find the cluster centers,
- a self-adjustable window length and
- an allele frequencies distribution plot to disambiguate ploidy estimations.

Removing extreme values

First we integrate a solution to deal with the non-significant data that over-extends the read count distribution range of the x-axis (see results of Magnolya on our simulated dataset in **Figure 12** and **Supplemental Table 3**). These are data points that for some reason (such as highly repeated segments, sequencing and alignment errors…) have unusually high coverage values that bias the depth analysis. When all of the significant data is compressed in a short segment of the x axis, it is difficult for the algorithm to identify the peaks of the clusters that correspond with the different copy numbers. By using only 99% of the bottom values coverage data, the abscissa gets reframed into the significant portion of the reads that carry the information we're interested in, and we get rid of potential extreme values without affecting the rest of the analysis. The data could be preprocessed and cleaned with this intention, but we implemented this step into our algorithm so that it would automatically perform an x-axis rearrangement whenever a very small portion of the data happened to have outliers. By reframing the x-axis, we enhance the portion of the data that has the clusters that define the copy numbers, making them more clearly identifiable and measurable. 99% is the default value but the user can change the parameters to fit different data sets.

The Sliding window

The second idea that characterizes PEDCA is that it measures the coverage over each contig with a sliding window that is considerably smaller than the contig length. While Magnolya evaluates the average coverage over the whole contig, PEDCA measures the depth at numerous points with a smaller sliding window of length *wl* that steps through the totality of the genome, each time

recording the number of reads that start at any of the positions within the window and averaging the count over the bin. This provides extra information about the composition of the assembly, allowing the detection of copy number changes and structural variations inside a single contig.

A parser browses the .sam/.bam file to recover the starting positions of each of the reads that has been aligned to the reference. PEDCA then runs a sliding window of length *wl* over the extent of each contig, stepping each time over *wl* /2 bp of the reference genome, and measuring how many reads start over each step. The read count r_i of all reads starting at each position i within each step is averaged over the window length *wl* so that for every window, its average read count is given by:

$$W_i = \frac{\sum_i^{i+wl} r_i}{wl}$$

By default, *wl* is set to 500 bp but the value can be parameterized for tuning over different contig sizes. We demonstrate the utility of this feature in the results section.

The resulting points are used for two purposes. First, by plotting the distribution of all possible read counts, we can observe the clusters that reflect the different copy numbers (**Figure 1 bottom**). Second, by plotting the coverage depth for each contig over the totality of the genome positions, we get a cloud of points $W_{i=0\rightarrow n}$ (**Figure 4**). Each point represents the average number of reads starting at that particular window step, and the mean value of the main clustered cloud reflects the ploidy/-ies of the sequence.

Inferring the read count cluster means by fitting a mixture of Gaussians

We first implement these improvements by following a very similar methodology to the one described by Van der Broek et al. 2015. We use JMEF (Garcia & Nielsen, n.d.), a Java cross-platform library who allows creating and managing mixture of exponential families. The code was adapted to work on bivariate data. We fit a mixture of Gaussians to the read count distribution with an Estimation Maximization algorithm to obtain the parameters of the mixture and, more specifically, the means of the clusters.

The sum of reads starting at each step, divided by the bin's length, gives us the average read count r_c of each window *w*. The average number of read counts in a region with a constant copy number depends on the number of reads, the length of the region and the length of the sliding window *wl*. The number of reads in such regions is approximately distributed according to Poisson, Po(λ), (Nter Klambauer et al., n.d.) where the mean number of reads per window is λ, given by:

$$\lambda = \frac{N\ wl}{G}$$

with *N* being the total number of reads over that region, *G* the size of the region and *wl* << *G*. Following (Xie & Tammi, 2009) we use the Gaussian distribution $\mathrm{Gauss}(r_c|\theta_{i,c})$ (instead of Magnolya's Poisson) to approximate the Poisson distribution with mean and variance $\lambda = \mu = \sigma^2$. This approximation is reported by Xie to work well when the mean number of reads per window is greater than 10.

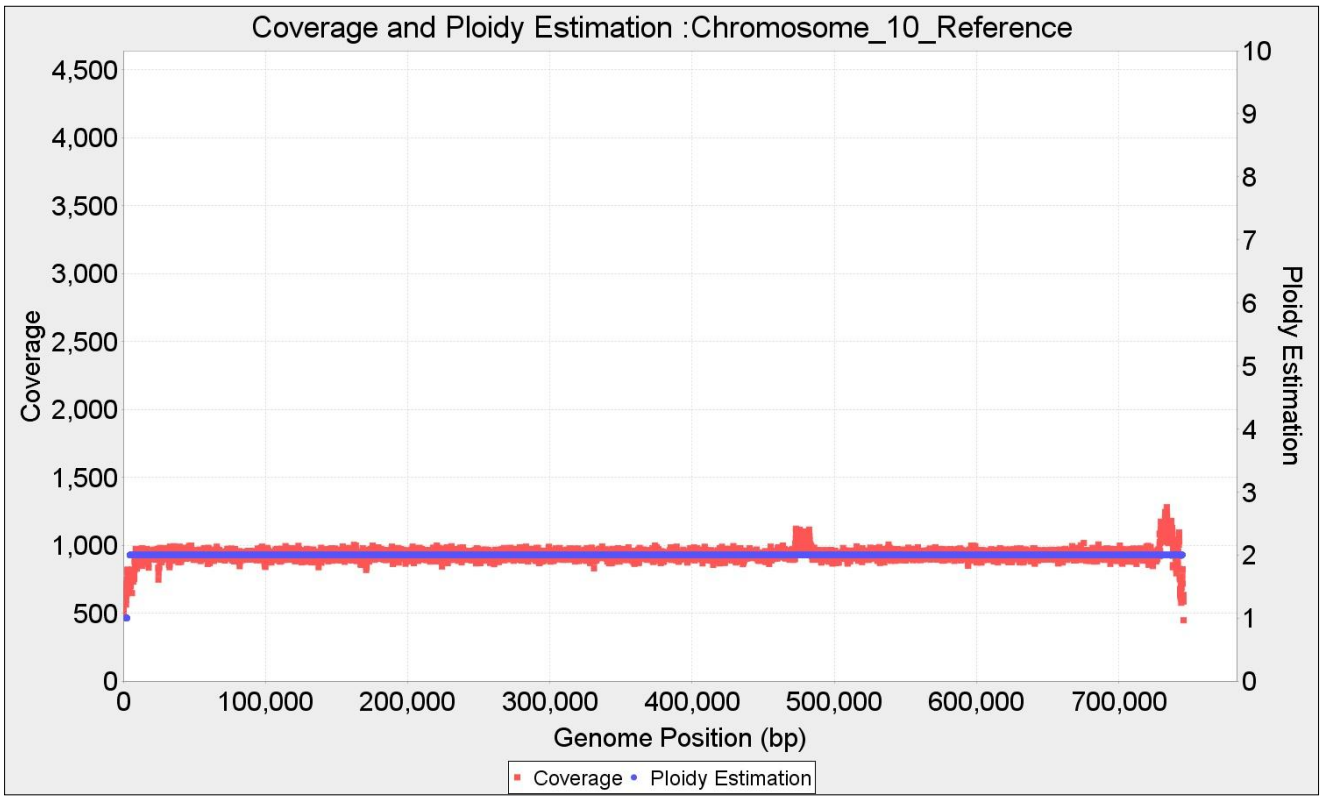


Figure 4 PEDCA outputs two different overlapped plots for each input contig. The first is a cloud of points (red) showing the coverage of each window sliding over the different positions of the reference genome. While the coverage tends to remain constant along the contig, some variations happen notably around areas with repeats or other coverage irregularities, often at the beginning and the end of the contigs (the telomeres usually have a high repeat content). It also happens that some reference genomes have fragments that have a different copy number from the rest of the genome as a result of scaffolding together different chromosomes; this can also easily be visualized in the contig coverage plot. The ploidy estimation corresponding to the coverage cloud overlaps in blue.

The read count distribution contains all the contigs with different copy numbers which together can be modeled as a mixture of *M* Gaussian components. Then, the probability of a certain read count r_c having a copy number of *i* can be expressed as:

$$p(r_c) = \sum_{i=1}^{M} \pi_i \, Gauss(r_c \,|\theta_{i,c})$$

The mixture coefficients, or weights of each Gaussian element is given by π_i and the Gaussian parameters $\theta_{i,c}$ is given by the mean $\lambda = \mu$ which we estimate by Expectation-Maximization (EM). We tried setting the value of $\mu = \sigma 2$, as previously explained, and found that the result didn't significantly change when $\sigma 2$ is instead estimated as an extra parameter in the EM, since the objective of the fitting is to obtain the values of the means. We used the JMEF java library for managing mixtures of exponential families to implement the EM.

EM is a method to find the maximum likelihood estimate of a set of parameters θ of a probability distribution, in this case: $\text{Gauss}(r_c|\theta_{i,c})$. It consists of two steps; an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters θ; and a maximization (M) step, which computes parameters maximizing the

expected log-likelihood found on the E step. The new θ are then used to determine the distribution of the variables in the next E step. The process is iterated until the maximum likelihood of the model does not improve beyond a certain threshold. The first set of parameter values are initialized with a K-means algorithm which clusters the read counts in *k* different initial clusters.

This approach present some challenges. The maximum copy number that PEDCA is able to detect is Max_Ploidy =10. It is relatively easy to determine the correct number of clusters present in the genome by visual inspection of the read count distribution, but we desire a process that is as automated as possible. We also want the algorithm to work with very diverse genomes. While it is easy to fix most of the parameters to properly identify the correct number of fits on our simulated data or any other particular single data set, we need to find ways to tune the settings to broaden the applicability of our method.

One of these parameters is the sampling rate of the read count distribution which affects the quality of the fit. More data points have higher standard variation and the algorithm tends to fit extra mixtures to try to reduce whatever error measure is used (**Figure 15**). In order to fix this step, it is necessary to find the right sampling rate; however, the correct sampling varies with the number of final mixtures (which we don't know in advance).

Another important challenge is finding an appropriate measure to fit the correct number of Gaussian mixtures.

A Naïve read count distribution smoother

One of the main reasons why the estimation maximization Gaussian fit failed to adapt to different data sets is that EM is not guaranteed to converge to a local minimum. It is only guaranteed to converge to a point with zero gradient of the likelihood function, which means that eventually it can get stuck in a saddle point. There are also many parameters involved into making the algorithm work correctly, such as the rate at which the read count is sampled; the range over which the read count number extends (the x-axis); the initialization of the Gaussians mixtures parameters in the EM algorithm; the measure to fit the right number of mixtures… All of these affect how the distribution is shaped and how easily its clusters can be identified. While it is relatively easy to set the parameters to solve a particular data set, it is more complicated to adapt the algorithm to work with a larger spectrum of genomes.

The final implementation of PEDCA opts for a simpler approach by skipping the fitting of the mixture Gaussian model. Instead, the read count distribution is smoothed in a naive way by a histogram of a fixed number of bins, converting the means of the clusters into the peak values of the function. Usually, even if the read count distribution can be described by a mixture of Gaussians, the real function is not a smooth curve but a sawed continuity of values that follows a certain local tendency. By applying a smoother, the only remaining maxima are the values at the peak of the clusters. When the function is smoothed in this way, detecting those points provides the means of the clusters. Those values have the same ratio among them as the chromosomal copy number, which we can then estimate.

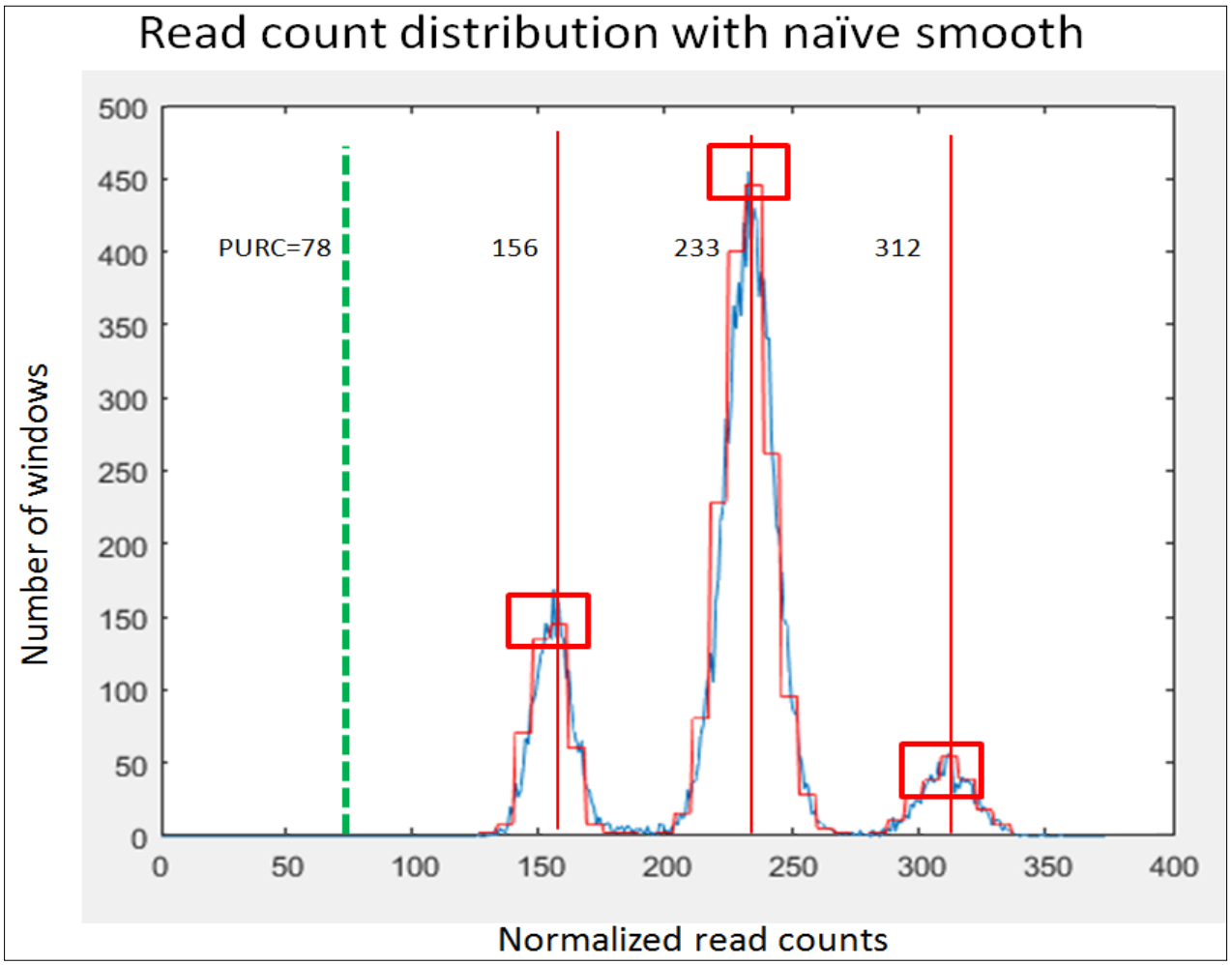


Figure 5 The original blue distribution with irregular values is smoothed by segmenting the function into bins and averaging the internal values. The result is shown in the red histogram. The red frames indicate the bins with maxima value. Searching back in the original distribution for the maxima value within those windows identifies the centers of each cluster (red dot lines at 156, 233, 312). In this example, the PURC (green dotted line) is identified around the value 78 and the existing copy numbers cluster around the ratios of 2x, 3x and 4x that value which, in turn, are the copy numbers found in this sample.

First we detect the limits of the x-values and decide whether or not it is necessary to reject the 1% that are outliers. This helps to delimit the region where the informative reads are, which allows the shape of the distribution to be more clearly defined and adapted to the fixed number of bins, and make the clusters easier to be identified.

We then proceed to plot the distribution as a histogram with cleaner tendencies. We divide the space of the abscissa in bins where some of them are turning points that we define as b_{max}, where $b_{max-1} < b_{max} > b_{max+1}$ Since we need at least 3 bins to identify a b_{max}, the bin width β depends on the histogram's domain and is inversely proportional to 3* Max_{ploidy}, so that it is defined by

$$\beta = \frac{\max\{r_c\}}{3*\text{Max\_Ploidy}}$$

with $max\{r_c\}$ being the maximum value in the x-axis. In most cases though, the different b_{max} won't correspond to both edges of the ploidy domain (1 and 10) at the same time. This means that most of the time we will need less than 30 bins. Therefore the default value is set to 2.5 with the possibility to be tuned by the user. We apply a naïve smoother by setting all of the values inside each bin to its average, resulting in a histogram of the read counts. Within each b_{max} the highest value of the original distribution is then retained as the mean of that cluster. This results in n means μ which are all approximative multiples of the Ploidy Unit Read Count (PURC) and have a ratio among them that can range between 1 and Max_{ploidy}. The PURC is the inferred read count of the basic cluster that serves as unit to all the others and is defined as the greatest common denominator of all μ (**Figure 5**).

The ratio model score

In order to infer the optimal PURC, a score is given to each of the ratios that could explain the ensemble of observed read count peak values as defined previously by $R=\{r_1, r_2, r_3, \dots r_n\}$. We can also define the ploidy read count unit as the hypothetical $r_0 = \rho$ that better explains all the members of *R*. The final ploidy estimation will be the ensemble $P=\{p_1, p_2, p_{3,} \dots, p_n\}$ defined by

$$P = \{\frac{r_1}{\rho}, \frac{r_2}{\rho} \dots \frac{r_n}{\rho}\} = R/\rho$$

Of course, we don't know ρ beforehand, but we know that it is a fraction r_1/j with ranges between $1 > j > Max_{ploidy}$. Since PEDCA deals with Max_{ploidy} = 10, it is computationally feasible to calculate all possible values of ρ_j. The ensemble R_j of theoretical potential clusters given by a candidate ρ_j is

$$R_j = \{1,2 \dots Max_{ploidy}\} * \rho_j$$

Not all of the possible clusters of R_j exist: only those that can be mapped to the actual peaks observed in R are retained. The values $r \in R$ that minimize the distance between themselves and the theoretical multiples of R_j give us the model explained by the candidate ρ_j

$$R'_j = argmin_{\,r \in R} \{\left|\left|R_j - R\right|\right|\}$$

Each model R'_j has a score σ_j which is given by the total distances between the observed clusters and the candidate model. The distances are normalized by the length of the contig *c*

$$\sigma_j = \sum \frac{|\left|R'_j - R\right||}{c}$$

At the end, the model with the smallest score σ_j is retained and the corresponding ρ_j assigned as the PURC.

A dynamic window length

Often, when analyzing several contigs of different lengths, the size of the sliding window is not adapted to the smallest contigs. This leaves some of them without enough coverage information to allow a copy number inference. This motivates us to implement a feature that allows the size of the sliding window to automatically adapt itself to the length of the smallest unsolved contigs in order to get enough data points and intent a new ploidy estimation round that adds extra results to the original round.

We've described how PEDCA runs a sliding window of length *wl* over the length of each contig to measure the depth of coverage at each step. A bigger *wl* provides a smoother averaged value of the read count for each window, resulting in a plot with less variations and more clearly distinct clusters. This might be very helpful when identifying the ratio between the clusters and the PURC.

On the other hand, a very high *wl* value provides fewer points to define the coverage of a contig. Some contigs might be even shorter than a certain window length that would otherwise work on the rest of the genome, which would mean that we wouldn't have any coverage points at all with which to estimate those minority contigs ploidy.

Thus, an adaptable window size allows examining the data from different angles, providing more accurate information at different levels of the analysis. This feature also enables to tune the algorithm to better cope with different genome characteristics like different contig sizes, repetitive content or coverage standard variation.

To address this, PEDCA runs a first round with the initial window length provided by the user or the default value, preferably a large value (depending on the size of the contigs it might be between 500 bp to 10 Kbp). If necessary, *wl* is dynamically adapted to solve the remaining contigs in a second round with a smaller size.

In its current implementation, the new window length is computed by keeping track of all contigs that could not be solved with the initial size, then running a second round with a new window based on the size of the smallest unsolved contig min {L}. The new window length is given by:

$$wl' = \frac{wl}{3 * z * min\{L\}} \qquad (1)$$

Where *z* is a parameter that insures that the ploidy estimation points are constant over a certain number of windows. This parameter *z* will be described in detail later (page **18**). *z* is multiplied by the constant 3 to ensure that the new window length provides at least three times more than the minimum required coverage points to estimate a ploidy on the smallest contig. The smallest new window length accepted is 16 bp.

While the current version of PEDCA runs by default with minimum with two *wl*, the quality of the obtained information can greatly improve by multiple runs with different lengths. An additional multirun option launches PEDCA with a preselected set of five window sizes {500,750,1000,2000,3000} providing a range of results that can be compared against eachother.

Solving ambiguous copy numbers: The allele frequencies distribution plot

The ensemble of different chromosome copy numbers that a genome display (or ploidy) can be described by an ensemble of integers $P=\{p_1, p_2, p_3 \ldots p_n\}$ Because our method is based on an analysis of the proportion of different depths along the contigs, every multiple of the basic ensemble describing P could describe the same read count distribution. The read count distribution clusters around Gaussian like bells. We define the ensemble of its read count cluster centers by $R=\{r_1, r_2, r_3, \ldots r_n\}$. For instance, read counts clustered around $R_1\{30,90\}$ can be described by the corresponding basic proportion of ploidies $P_1 =\{1,3\}$, but also by {2,6} or {3,9} and any other of its multiples. PEDCA only considers the smallest integer proportion of ploidies possible, so only the basic explanation $P_1 =\{1,3\}$, would be retained. The PURC is the number of read counts r_0 corresponding to the basic cluster center that best explains the ploidy $p_0 = 1$ (**Figure 5**).

The precision of the ploidy estimations depends on the shape of the read count distribution and, more precisely, on how distinct its clusters are and how much deviation they contain. For diverse reasons that will be described later, the ratios among the clusters centers can sometimes be measured with some imprecision, which might lead to a wrong estimation of the PURC by a fixed ratio. When this ambiguity arises, the wrong estimation differs usually by a ½ f the real ratio, but it can also be $^1/_3$ and theoretically up to ¼. When necessary, PEDCA outputs a distribution of the allele frequencies contained within the first two clusters in order to allow an analysis and solve the ambiguities.

The window size might also affect the smoothness of the read count distribution. When the function is too irregular or spiky, the smoother might wrongly classify some peaks as a separate cluster.

A similar complication might arise with clusters of high ploidy. Each cluster builds around a multiple of the PURC, and follows a Gaussian distribution. The sums of Gaussian distributions have the property that each time they add up, their mean and standard deviation also adds up. A cluster of μ=5 and σ=2 will result in a twofold multiple of μ=10 and σ=4. This has more effect on the clusters with high copy numbers. The standard deviation of their distribution might overlap their tails, potentially displacing their centers to each other, and taking them out of alignment with the corresponding theoretical multiple of the PURC. This might result in a wrong estimation of the PURC and therefore in an erroneous ploidy estimation (**Figure 6**). The added variation reflects also on the coverage plots, with depth point clouds being more dispersed as the ploidy increases (**Figure 7**).

In order to disambiguate these difficult ploidy estimations, we added an extra feature. If provided with an input variant call file (.vcf), PEDCA measures the proportion of the allele frequencies in those contigs with ploidy corresponding to the first cluster. Those contigs have the smallest ploidy possible (p_1) of the whole genome and are assumed to be $1 \leq p_1 \leq 4$. The shape of this distribution provides additional information about the ploidy of these contigs (**Figure 8**). A diploid contig will have only one peak, since most variations will have only two alleles, the shape of the proportion will cluster around ½. A triploid will show most proportions around the 0.33 and 0.66 values (1/3 and 2/3). Tetraploids will show three peaks around 0.25, 0.5 and 0.75 values.

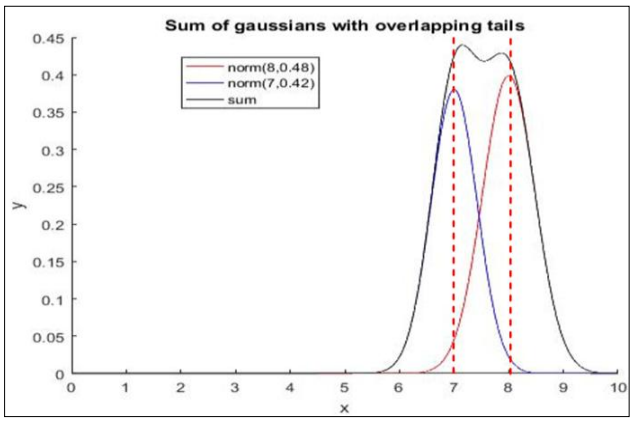


Figure 6 The curve resulting from adding two Gaussians with overlapping tails might offset the center of their means. Instead of μ_1=7 (blue normal) and μ_2=8 (magenta normal), the peak of the sum of Gaussians (black) is offset towards +/- μ_1 =7.2 and μ_2 =7.8 This ends up potentially affecting the inference of the ratio unit (PURC).

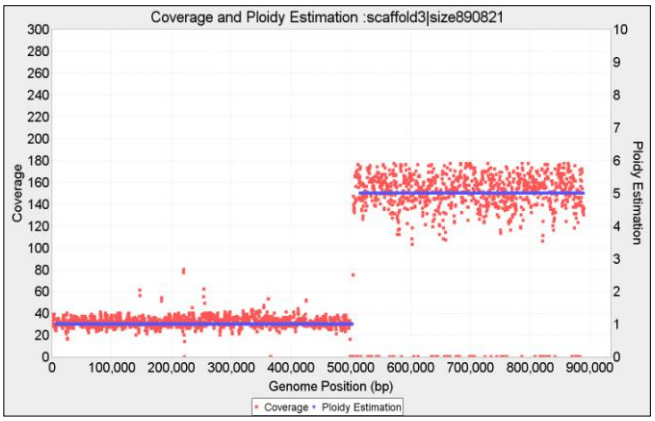


Figure 7 The standard variation doubles each time that the basic Gaussian distribution is added. This effect can also be visualized in the coverage plot: from p=1 to p= 5 the coverage data points get much more scattered around their mean. This is a limit of any depth of coverage approach, which cannot correctly identify high copy numbers.

In order to do this, PEDCA stores all contigs that have an estimated copy number corresponding to the first identified cluster. Because some contigs might be aligned to chromosomes that have structural variations within them with different copy number, only those contigs with contiguous ploidy estimation are stored. This leads to less noise in the base call distribution plot.

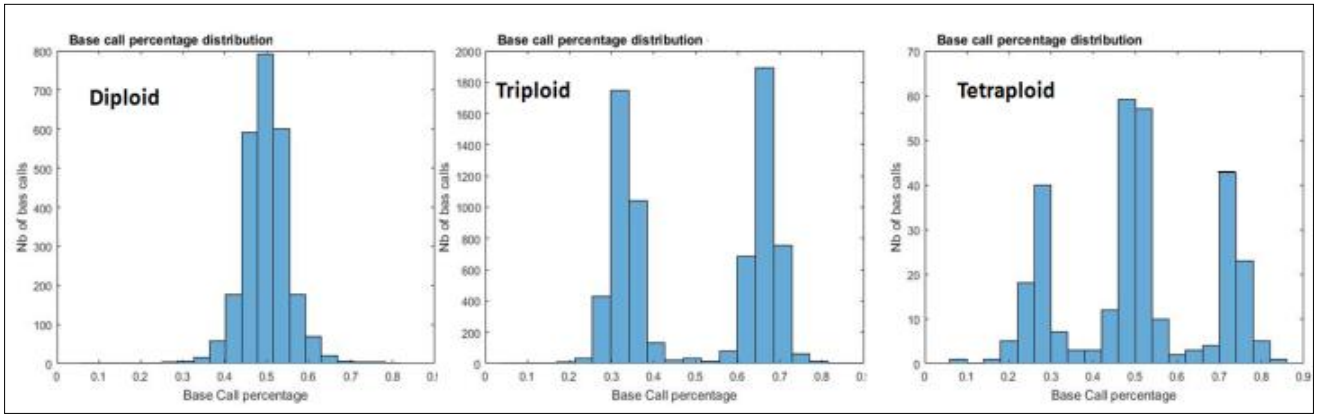


Figure 8 Allele frequencies have typical recognizable shapes that reflect the ploidy of their contigs. Diploid (left figure) are built around 0.5 value; triploid (center figure) around 0.33 and 0.66 values; and tetraploids, (right figure) around 0.25, 0.5 and 0.75.

Before plotting, PEDCA removes the base calls above 90% and below 10% to ease the read of the shape. We consider that all first clusters will have a ploidy between 1 and 4, and therefore those variations are considered errors or non representative.

The plots analysis provides an extra element of information to the ploidy inference. The base call of a contig with p_1=1, doesn't have a particularly recognizable shape because only a small set of random variations or errors will be registered in the .vcf file. For this reason, we output also the second cluster distribution p_2, which will always be $p_2 \geq 2$

The output

PEDCA outputs two overlaid plots for every input contig. The first is a cloud of points showing the coverage of each window sliding over the different positions of the reference genome. While the coverage tends to remain constant along the contig, some variations might happen, especially around areas with repeat content. It also happens that some reference genomes have sections that have a different copy number from the rest of the genome; this can also be easily visualized in the contig coverage plot (**Figure 9**).

The second plot that PEDCA outputs for those contigs which have sufficient coverage data, is the corresponding copy number estimation over the coverage cloud. Each ploidy estimation point pi is computed by rounding the ratio of its corresponding coverage read count point ri over the PURC.

All plots and charts are produced using JFreeChart 1.0.19 (David Gilbert, n.d.).

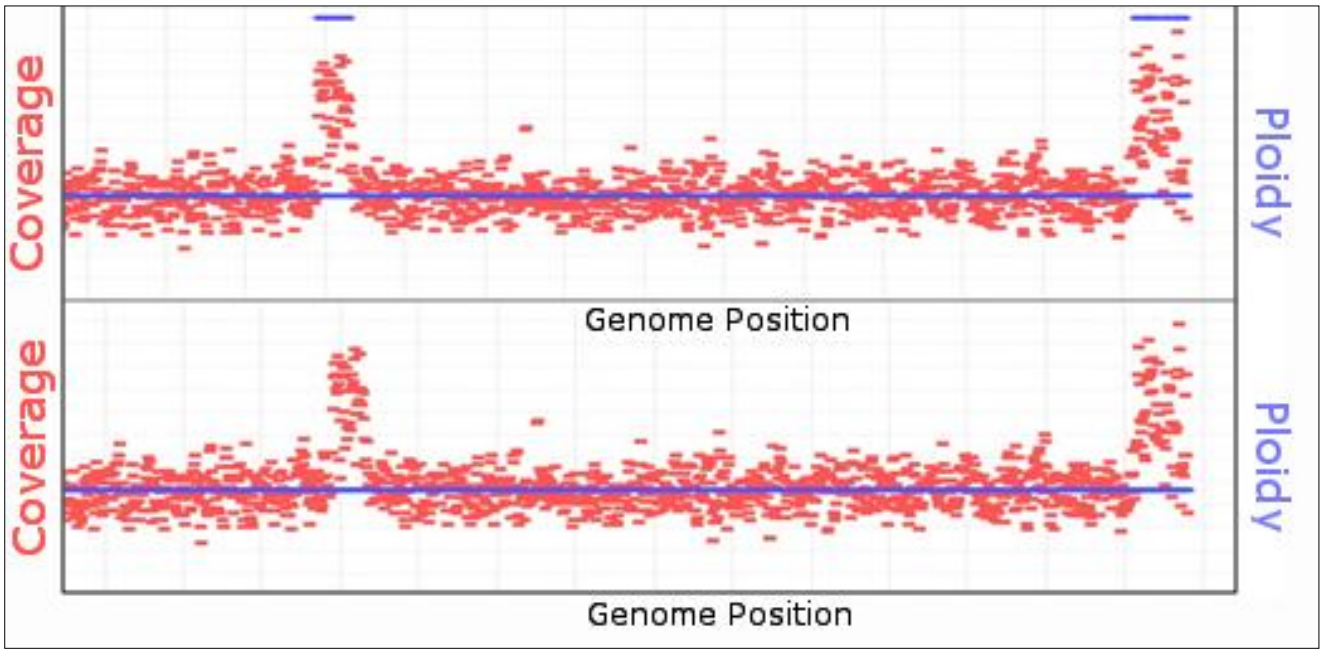


Figure 9 Computing the mode over different k contiguous ploidy estimation points. If k is too small, it might lead to fragmented ploidy estimation in regions with noisy coverage (top). The continuity is smoothed with the right k value (bottom k=50). The correct length of k depends on the required precision, and can be parameterized. If k is too big, it might lead to the non detection of regions with different ploidies (i.e. large structural variations found in hybrid genomes).

The ploidy estimation plot reflects the same irregularities that the coverage plots might have. To help understand the sense of the dataset we need to smooth the variation in the ploidy estimation points. The ploidy estimation of a segment is given by a continuity of coverage points divided by the PURC, resulting in an integer. Because of the variation in depth along the genome, there might be oscillations between different values, resulting in copy number jumps (**Figure 9 top**).

Among the seemingly random population of points, we are looking to retain those that appear consistently around some average. But simply averaging them over *k* different positions doesn't always work because the average might be a point that doesn't reflect any real ploidy present in the genome. For instance, a contig with copy number fragments of 2 and 4 copies might average erroneously to 3 over some regions.

Instead, we choose the mode of the ploidy over *k* contiguous windows, with *k* being a parametric value that defaults to 50 windows. There is a tradeoff between eliminating small variations to reduce noise in the interpretation of the ploidy and being able to identify fragments with real copy number jumps. For the estimation report, it is important to know the size of the fragments with different copy number that PEDCA is able to detect.

If we are retaining only the mode value over *k* points, then the minimal number of occurrences that the mode can have is *k* divided by all possible values of the ploidy (= Max_{ploidy}). Fragments with fewer occurrences will not be detected and this information is reported in the results file. Another way of explaining this is that the worst precision γ of the ploidy estimation that can be measured occurs when the dispersion over the *k* points is the highest. If no ploidy is dominant along the bin, we can assume that the mode is supported by at most $1/Max_{ploidy}$data points. Since each data point is taken every window length *wl/2* :

$$\gamma = \frac{k*wl}{2*Max\_Ploidy} \qquad (2)$$

It is important to underline that this theoretical worst case precision relies on the accuracy of the coverage data. But the quality of the coverage data can also be affected by other factors. When the window length is too small, the coverage reported by the alignment in regions with a higher repeat content will have a sudden increase in their coverage. Small window sizes will result in more data points defining a given region. Other regions with high variation rate might overrun the capacity of the aligner to map the reads to the appropriate draft reference, resulting in improper reporting of coverage rates. In those cases, even if the mode reflects a ploidy estimation value, it will not correspond to a real copy number change. This is why γ can also be given by the user and adapted to the specificities of the genome.

PEDCA outputs a .txt file describing in detail the different regions of each contig that have different ploidy estimation. We still might face some noise due to non continuity within some contigs. Before deciding that a given fragment really has a change in its ploidy, PEDCA makes sure that at least a certain number of continuous data points bear the same copy number estimation. This intends to avoid erroneous oscillating estimations in inconsistent data sets. If the data does not ensure enough continuity, PEDCA breaks the contig into fragments and outputs an estimation only for the segments for which it has enough evidence. By default, that number is *z*=20 coverage points, but can also be defined by the user. Only fragments with continuous ploidy over more than φ bp will be reported.

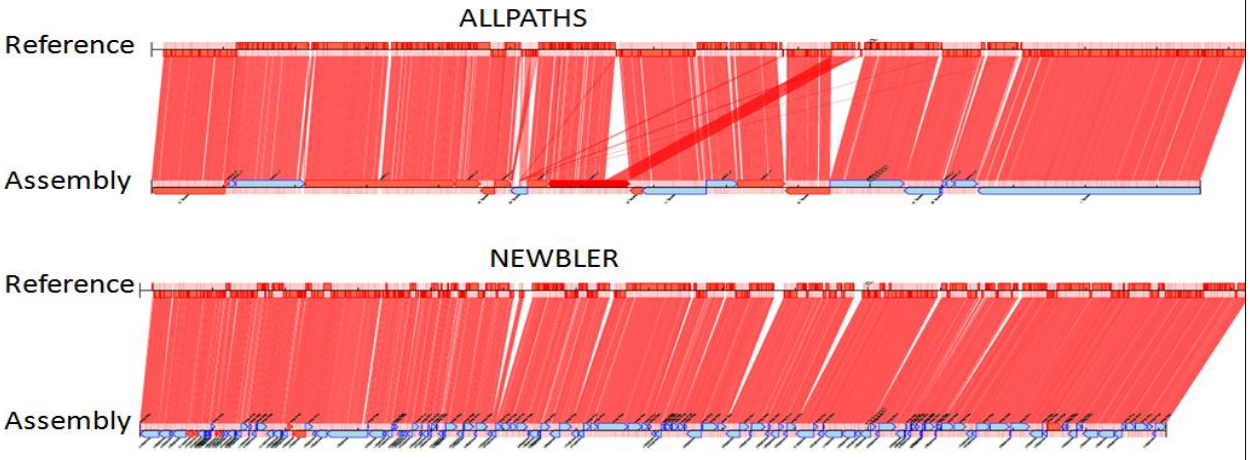
$$\varphi = \frac{z*wl}{2} \qquad (3)$$

Assembly tests and Ploidy Results

We mapped the results of our assemblies of simulated reads to their known references, analyzed the errors with Quast (Gurevich, Saveliev, Vyahhi, & Tesler, 2013) and visualized with Contiguator (Galardini, Biondi, Bazzicalupo, & Mengoni, 2011).

ALLPATHS managed to obtain 61 contigs with a total of an N50 of 133 Kbp. Newbler's assembly resulted in 558 contigs, almost 10 times more scatter than ALLPATHS'. Its contigs were considerably smaller in average with an N50 of 10.8 Kbp, but with a remarkable absence of any assembly's errors (relocations, translocations and inversions, **Figure 10**). A SOAPdenovo assembly was also performed with less satisfying results.

Since our simulated dataset is very close to the real *pastorianus* genome that we intend to solve, we prefer the Newbler assembly with zero misassemblies over ALLPATHS. Even if ALLAPATHS manages to map 5% supplemental sequence, it does so with assembly errors that we prefer to avoid. We prefer this choice because PEDCA is able to identify fragmented ploidies over the extent of the contigs. If the provided reference is wrongly scaffolded, ploidies from different chromosomes might appear together in the same contig leading to false structural variations assumptions.



Mapping stats for all chromosomes

	Contigs	Mapped Contigs	N 50	Misassembles
ALLPATHS	61 - 3.5 Mbp 96.6%	56 - 3.4 Mbp 95.3%	133 Kbp	26 - 856 Kbp 23.5%
Newbler	558 - 3.3 Mbp 91.2%	507 - 3.2 Mbp 90.1%	10.8 Kbp	0

Figure 10 A visual comparison between ALLPATHS and Newbler assemblies of our simulated chromosome 4. The top horizontal sequence of each diagram represents the reference genome; the bottom line shows the contigs; red lines map the contigs to the correct position in the reference (visualization with Contiguator) It shows that ALLPATHS have clearly longer contigs (bottom horizontal line) and a slightly more dense coverage of the reference genome (top), but Newbler doesn't have misassemblies. The table describes the results over all chromosomes combined. It is worth underlying that, while the N50 is much lower with Newbler, the mapped reference genome is only slightly affected while the misassemblies drop drastically to zero.

With Quast, we measured the optimal parameters on the Newbler assembly to minimize the different errors and amount of unmapped contigs (**Supplemental Figure 3**). Comparing the different assemblies from the three programs revealed common unmapped areas, regardless of the software used. We found that the optimal parameters, confirm the settings used by (van den Broek et al., 2015). Nevertheless, it is important to underline that these tests were performed to get to know our data better. Our method is completely independent from any particular assembler.

The quality of the input draft reference genome affects the quality of the ploidy estimation in different ways. Error-free contigs with a perfect alignment of the reads to its reference would yield the best ploidy estimations with minimum false ploidy fragmentation, while low quality assemblies might merge sequences coming from different chromosomes that will lead to sudden jumps in ploidy estimation that do not reflect the real structure of the chromosomes. The quality of the input draft reference is therefore the user's responsibility.

Ploidy Estimation Results on simulated data

We run ConPADE with 4 simulated library sizes, and we measure the likelihood for plausible ploidies from 1 to 4. ConPADE correctly predicted 4 out of 5 chromosomes in our simulated dataset. Their predictions were constant; the estimation for each chromosome was always the same regardless of the library used. For chromosome 10, it misestimated a copy number of 4 instead of the correct 2. We also noted that the estimated likelihood for different ploidies were often very close to one another, which could easily lead to potential estimation errors (**Figure 11**). ConPADE considers, at most, only two possible alleles at any given position.

While the underlying principle of Magnolya is quite solid, when the software is run on our simulated data set it is unable to detect the right chromosome copy numbers. Apparently, the three peaks that are supposed to be identified are in the read count distribution (**Figure 12**), but the x-axis is so stretched that only the biggest central peak is recognized while the two others, being too close together, are discarded. Magnolya seems to have a problem when the x-axis spreads over a large range, but most of the values are concentrated in only a small portion of that range.

We also estimate the ploidy of our simulated data by fitting a mixture of Gaussians to the read count distribution. We fix the sample rate to 40 points, which results in an appropriate sampling for our data set. We then run 10 fitting rounds, starting with 1 Gaussian and increasing each time the number of mixtures. We then compare all of them and try to determine the point where adding an extra Gaussian no longer adds significant information to the distribution curve.

We compare different methods to determine the right number of mixtures to be fitted. The right measure should improve its score while the right number of clusters has not yet been reached (in this case, 3), but should stop improving with subsequent added Gaussians.

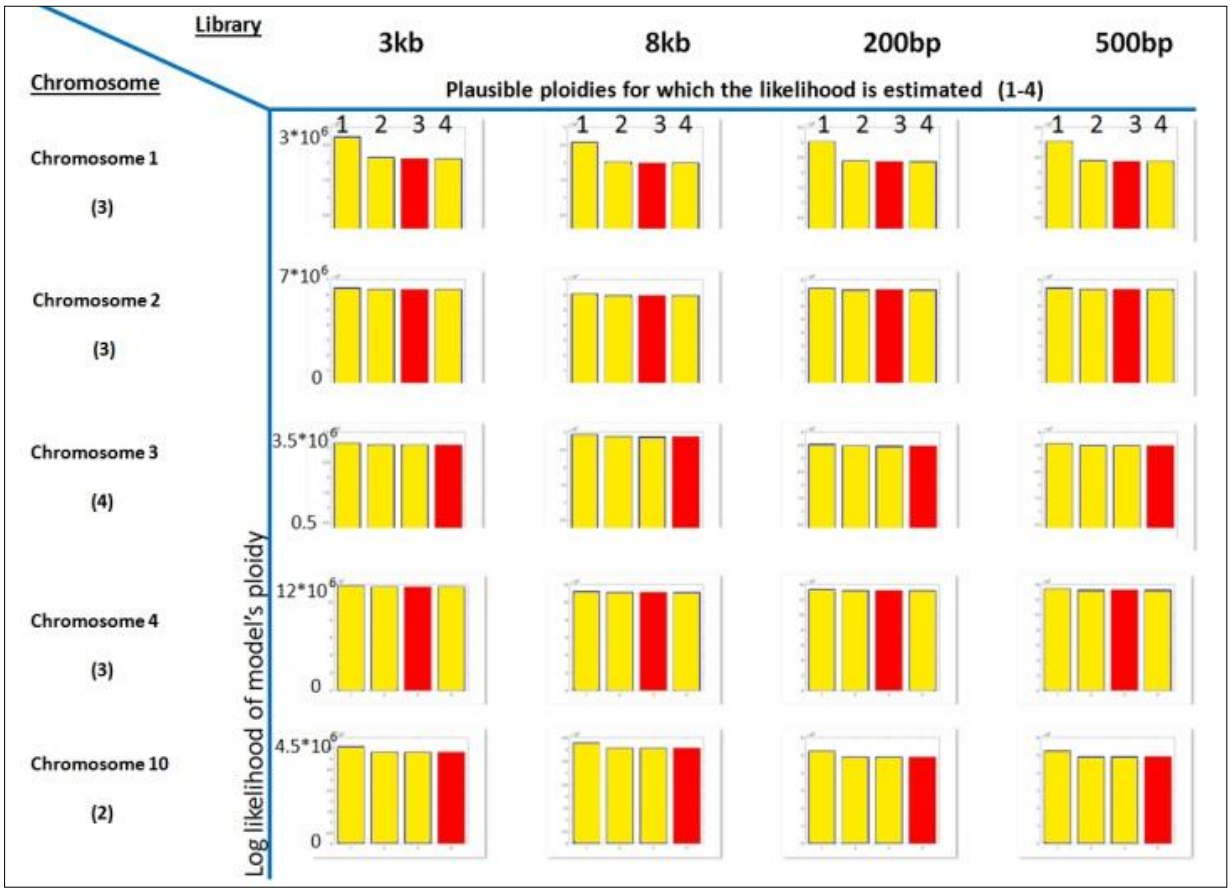


Figure 11 ConPADE ploidy estimation results on simulated data. On different library sizes the results were similar. For each chromosome, the right copy number is indicated in parenthesis. In red, the lowest log likelihood (ConPADE's prediction) for different possible copy numbers (1-4). Chromosomes 1-4 were correctly estimated. Chromosome 10 was predicted to have 4 instead of 2 copies.

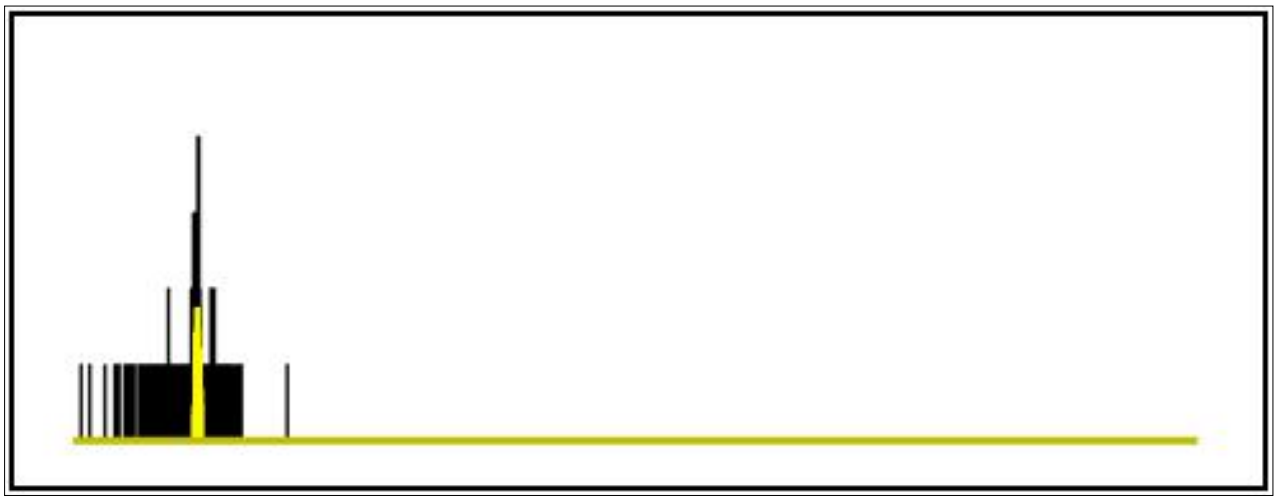


Figure 12 Magnolya's fit of Poisson mixtures to the simulated dataset. Only the main peak (yellow) is identified, consequently, a copy number of 1 is wrongly identified for the whole dataset. The range of the x-axis is too large and all the clusters of the read count distribution are compressed in a relatively small portion of the plot which doesn't allow Magnolya to correctly fit the right number of Poisson mixtures.

While the Sum of Squares due to Error (SSE), the R-square (R-Sq),the adjusted R-square (adj R-Sq) and the degrees of freedom in the error (DF) are good measures for the fitness of the mixture to the data, they don't serve our cause well enough since they don't always halt the addition of new mixtures to detect the right number of clusters (**Figure 13**). The algorithm keeps fitting mixtures beyond the main number clusters and reducing the measured error to the existing points. Unfortunately the extra mixtures add nonexistent centers that prevent finding the correct ratio among the real ones. These measures were discarded as they failed to fit the correct number of mixtures (*i*=3) in our simulated data set in more than half of the time.

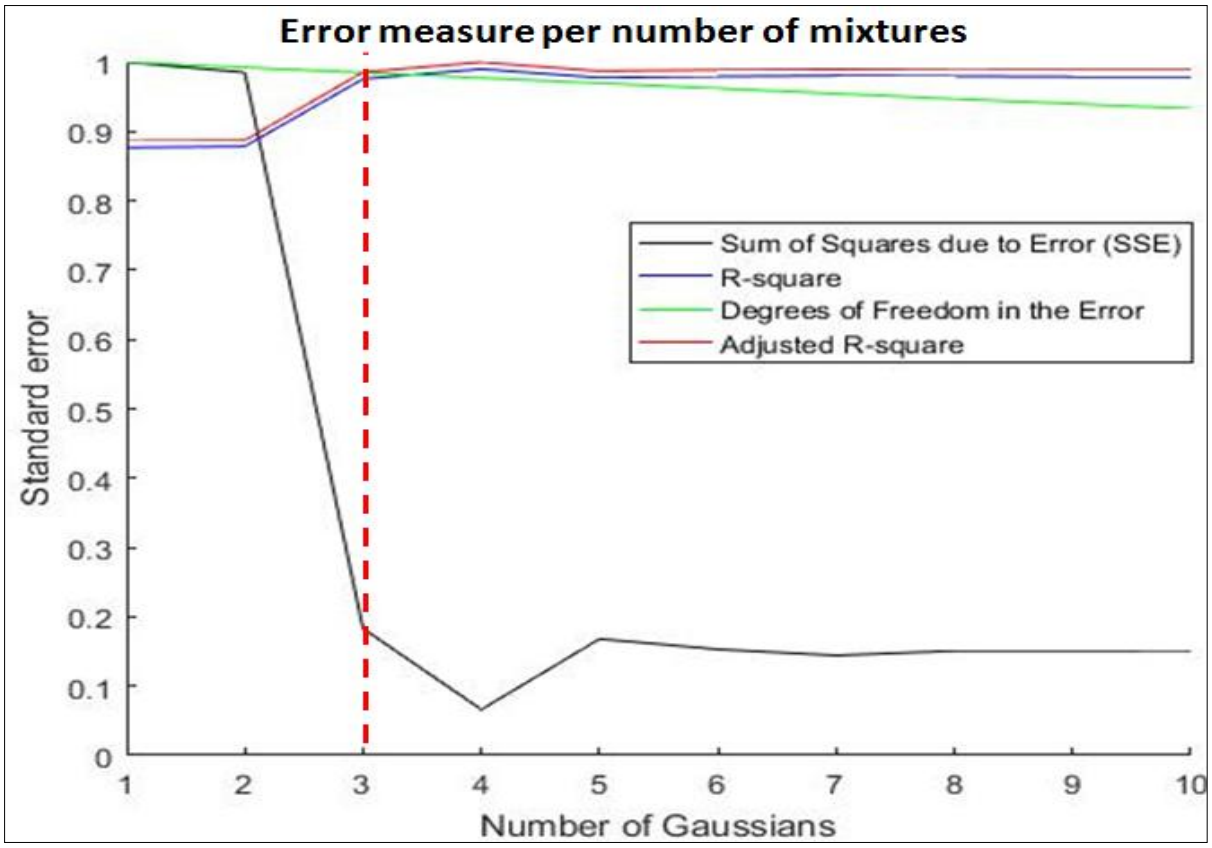


Figure 13 A comparative study of different measures to fit the number of Gaussian mixtures. When fitting a curve to our simulated dataset (which should stop at 3 clusters – vertical red dotted line-), most of the measures show an important improvement at the expected number, but all of them better explain the model by overfitting an extra fourth mixture.

(Nijkamp et al., 2012) use the Bayesian Information Criterion (BIC) to estimate the right number of fits because it penalizes each extra parameter and tends to avoid over-fitting the distribution. We measured how well the BIC worked on our simulated genome. With the EM approach we fit 10 different models to the read count distribution, each of them with different number of Gaussians 1-10. The model with lowest BIC is then selected. We run our algorithm 1000 times, which correctly predicted the copy number of the genome 82% of the time. This gives a better result than any of the error measures previously described. Because the initialization of some of the EM parameters is random (the original μ's are initialized with k-means), we might have slightly different results each time that the algorithm is run (**Figure 14**).

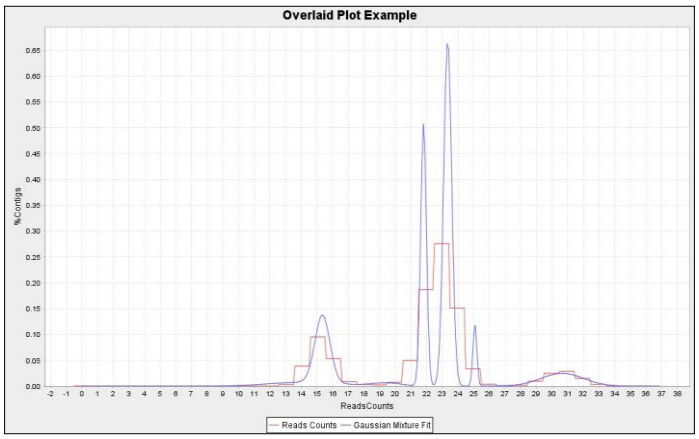


Figure 14 Gaussian mixture fit of the simulated genome read count distribution. The blue curve fits the red read counts histogram with the BIC measuring when to stop adding mixtures. The result is not constant, fitting in some rare cases up to 8 mixtures (instead of 3), making the method unreliable to detect the clusters centers.

We decided to average the results of several runs (100) in order to reduce the impact of the occasional wrong predictions. With this method PEDCA was able to accurately predict the copy number of our simulated dataset in 97.2% of the cases. Unfortunately the results dropped to 64.1% when real data was used.

We also noted that the sampling rate had an impact on the performance of the fit. When too many points are sampled, the method tends to fit extra mixtures to reduce the error of data points that were not covered by the main mixtures. This complicates the task of identifying the right number of clusters (**Figure 15**).

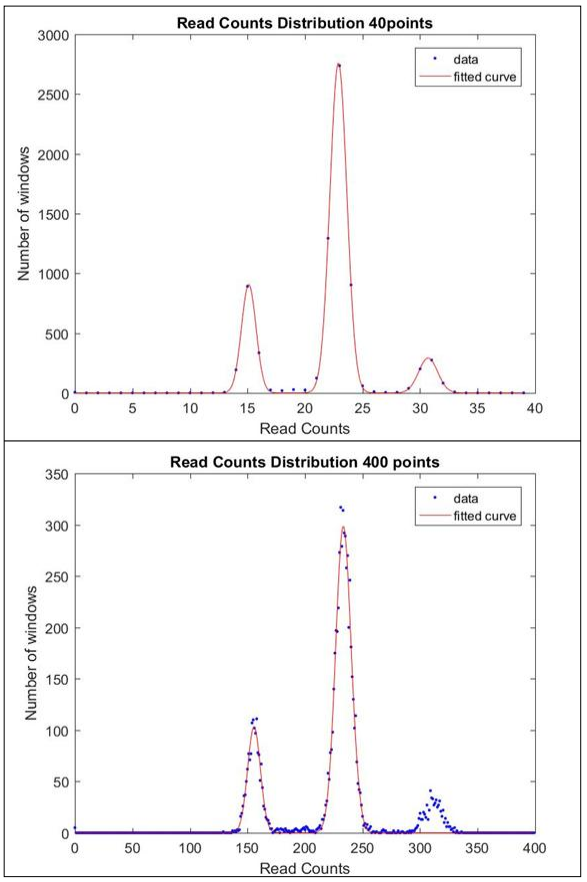


Figure 15 Small sampling in the read count distribution leads to a function which is easier to fit. But finding the right sampling in advance might be a challenge. 40 points offer sufficient sampling when only 3 mixtures must be fit (top figure), but it doesn't provide enough numbers for a high number of different ploidies. On the other hand, when too many points with a large variation exist, the EM algorithm might fit extra mixtures that don't explain new clusters but include unexplained points. In some runs with 400 points (bottom figure), three mixtures were used to fit two big clusters. Means derived from such a fit cannot be used to infer the centers.

PEDCA results on the simulated genome

After aligning all of the simulated reads to the simulated genome (the first copy of each chromosome is used as reference) with bwa-0.7.13, we obtain the .bam file that we use as input. PEDCA's results on our simulated genome are very promising. PEDCA correctly predicts the ploidy of each chromosome with the default window length. The read count distribution has clearly distinct clusters over a very large range of different window sizes, including very large and very short windows (**Figure 16**).

The naïve smoother successfully fits and identifies the different clusters even with very large *wl*'s. Because of repeated regions and other coverage irregularities over the sequence, a small window length sometimes leads to breaks in the continuity of the coverage data points, leaving the corresponding ploidy estimation points blank. PEDCA reports only those segments for which it has information, reporting their starting and end points, even if this leads to a more fragmented report.

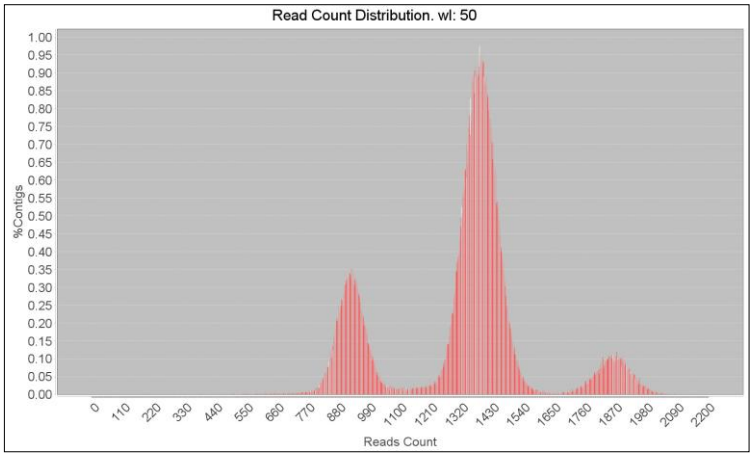


Figure 16 With our simulated genome, very distinct clusters are obtained even with a very small window size of 50 bp.

We evaluate the precision of the ploidy estimation by measuring the proportion of the genome (3.6 Mbp) that is correctly estimated and the total area that has any estimation output at all (correct or not). We are interested in how the main parameter -the window length- affects the performance (**Figure 17**). The blue histogram shows that there can be minimal variations in results from very small window sizes (50 bp) to very large ones (100 Kb) and that the vast majority of the genome's ploidy is correctly predicted. All of the lengths correctly predict more than 99.5% of the genome: and the best results are obtained with a window length of 600 bp which correctly predicts 99.98%. Let us remember that a small Structural Variation (SV) of 10 Kb was introduced in Chromosome 3, giving a different copy number to one of the alleles. This is the 0.27 % of the genome that remains unrecognized with window sizes bigger than the threshold φ calculated with (equation 3 page **22**).

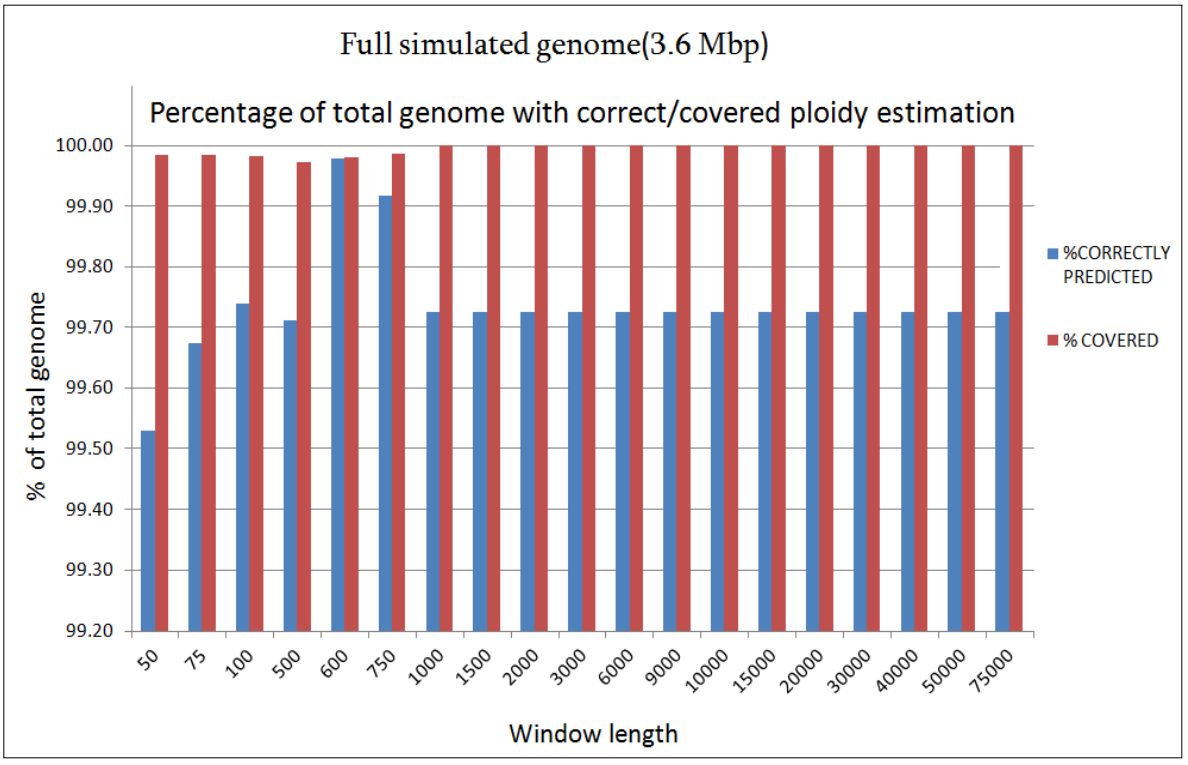


Figure 17 Percentage of all of the simulated genome with ploidy estimation and percentage correctly predicted under different window lengths (note that the y-axis starts at 99.2%). More than 99.98 of the genome had an output estimation (red) regardless of its accuracy. All window lengths detected at least 99.52% of the genomes ploidy correctly (blue). With sizes beyond the threshold that can detect the 10 Kbp deletion introduced in Chromosome 3, the percentage of correctly predicted genome remains constant since the SV is no longer recognized.

Because some regions contain coverage data that is difficult to analyze or contain no data at all, they might not contain an estimation over certain areas. Therefore, we are also interested in evaluating the ratio of each chromosome that is covered by a ploidy estimation, whether it is a correct or erroneous one. The red histogram in **Figure 17** shows that with windows larger than the deletion (10.000 bp), all the genome is covered despite the region of the structural variation being unrecognized. With shorter windows, only a small amount of fragments (+/- 0.02%) are not covered by the estimation. The difference between the covered region (red) and the correct prediction (blue) corresponds to the segments that are wrongly predicted. This value ranges between 0.48% with the smallest *wl*=50 bp, and ≈0% with the optimal *wl*=750 bp. The constant 0.27% difference after the φ threshold corresponds exactly to the SV which means that, except for that deletion, the rest of the genome is correctly predicted.

It is also interesting to measure the sensibility of the ploidy estimation by fragments. The simulated Chromosome 3 was built with a copy number of 4 but with a small SV in one of its haplotypes consisting of a 10kb deletion (from 306 Kbp to 316 Kbp) a few base pairs before the end of the sequence (**Supplemental Figure 2**).

We measure how much of that segment is correctly detected and estimated. The theoretical precision of the ploidy estimation by fragments depends on the window length, as defined in the Methods section. But the practical precision depends also on the reliability of the coverage. The fluctuation in coverage can be the result of something other than a change in copy number. They might reflect a segment that is highly repeated over the genome and show higher or lower depth than the reference fragment really has. This is often the case in the regions at the beginning and end of the chromosomes. Otherwise, it might also reflect less coverage than the one expected, for instance, if the region has more variations that the one that can be recognized by the alignment tool that generated the .bam file.

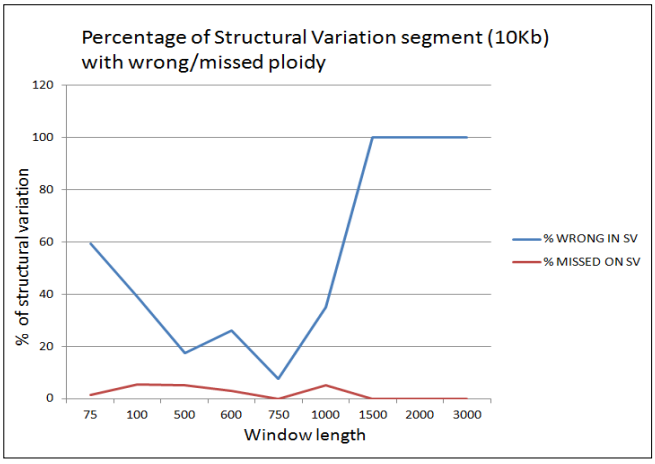


Figure 18 Percentage of the introduced Structural Variation introduced in Chromosome III that went undetected by Pedca (red) or had a wrong estimation under different window lengths (blue). Under the threshold that allows the detection, *wl*= 750 bp has most of the deletion correctly identified. Very small *wl*'s wrongly predict the ploidy of the SV, mostly because the coverage is highly fragmented. The smallest window provides too many points with an important amount of dispersion that increases the fragmentation of the ploidy estimation. A *wl*= 750 bp smoothes the dispersion by averaging the coverage over larger segments. Above *wl*=1000 bp the SV is not detected.

We measured the effectiveness of different window lengths detecting the SV introduced at the end of Chromosome 3 of our simulated data, which is a region very prone to coverage irregularities (**Figure 18**).

The results shows that window lengths smaller than 1.000bp are able to detect from 42% up to 92.5% of the deletion fragment, depending on which window length is used. The best result is obtained with *wl*=750 bp, which correctly identifies 9.250 bp of the deletion without missing any parts of the total fragment. A very small window length of 50 bp gives too many data points with too much standard deviation, which leads to an irregular ploidy estimation that only detects 42% of the fragment. With windows higher than 1.000 bp, the deletion is

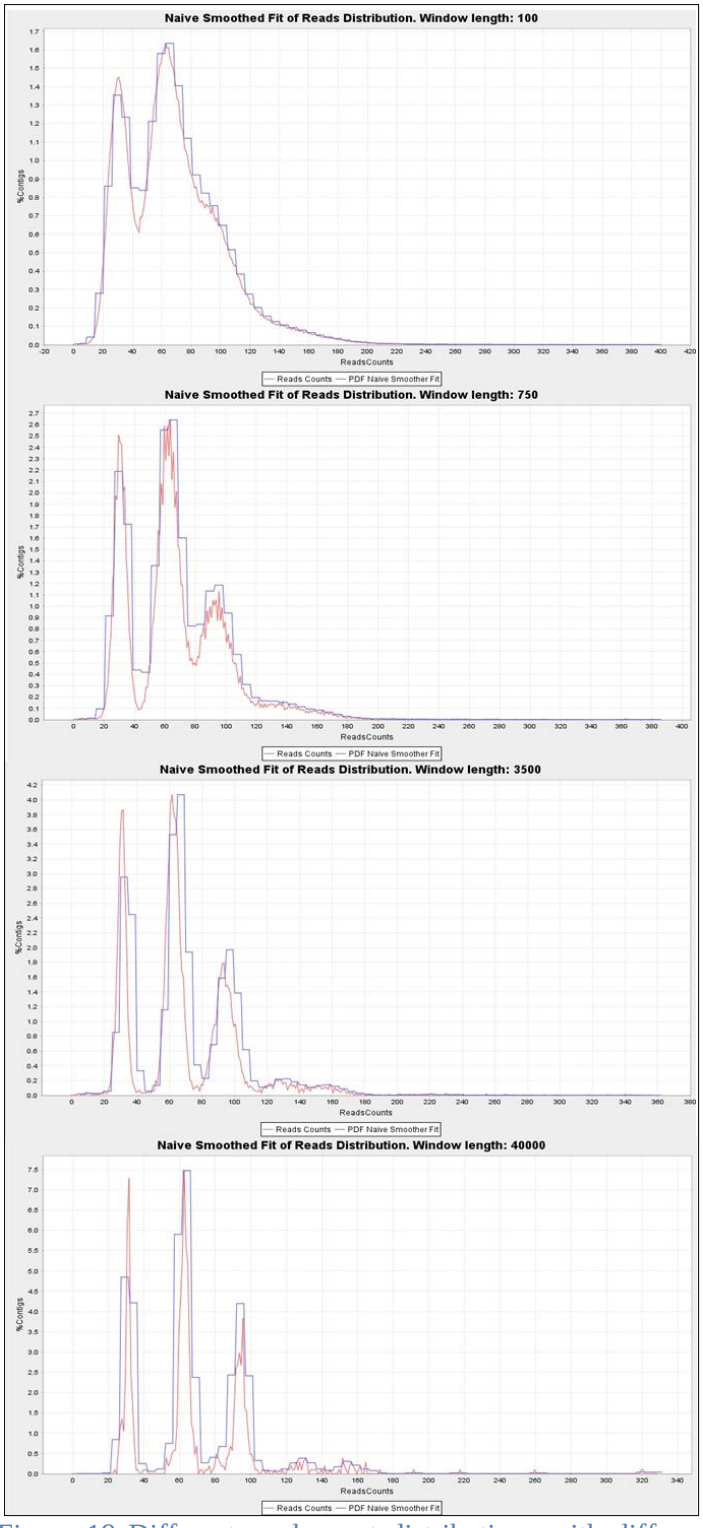


Figure 19 Different read count distributions with different window lengths on the Baseclear sequencing. From top to bottom: *wl*=100, 750, 3.500 bp and 40 Kbp. The shortest windows have too much standard deviation, joining clusters with different copy number to the point where they cannot be recognized anymore. As the window size increases, the clusters become more subtly defined and recognizable

beyond the φ threshold that allows it to be identified.

Results on the CBS Baseclear genome.

Our simulated genome results serve only as proof of concept. The simulated data is much cleaner than a real data set. Such neatness is rarely present in a real genome, but we can be confident that the principles on which PEDCA operates are sound.

We then apply our method to two real data sets of *S.pastorianus* CBS1483: the Baseclear CBS1483 assembly and Novogene CBS1483 (that could not work with Magnolya and we expect to be a very noisy assembly). The *S.pastorianus* CBS1483 genome that is used as reference in both cases consists of 59 scaffolded contigs with a total length of 22.3 Mbp (van den Broek et al., 2015). While all of the contigs are bigger than 500 bp, the average size is 378.8 Kbp with the largest being 1,464.6 Kbp and the smallest 526 bp. The N50 is 750.1 Kbp

On the Baseclear sequencing, windows smaller than 500 bp lead to very ambiguous groups in the read count distribution, where only two clusters can be recognized once the histogram has smoothed the tendencies. The smaller windows lead to less distinct clusters with a bigger amount of overlapping tails. Yet, PEDCA can operate if it correctly identifies at least the first two clusters of this sequencing P={1, 2}. Once the PURC is correctly inferred, the rest of the process can estimate all contig copy numbers by computing its ratio relative to the coverage (**Figure 19 top**).

Using the window size that best worked on our simulation (wl=750 bp), we obtain a read count distribution that displays 3 clusters (**Figure 19 center**) but they are less clearly distinct in the Baseclear sequencing than in our simulations. The reported ploidies are P={1, 2, 3}. The groups join their tails, especially the second and the third ones,

which indicates that a certain amount of regions will have ambiguous coverage. The last clear cluster has what falsely appears to be a long tail on its right side. The 43 contigs solved in the first round report 63 fragments of different ploidy estimations. This is one of the least fragmented reports (77 fragments), which is very close to the best one (wl=500bp; 71 fragments).

Here, the adaptable window feature proves to be very helpful since the second round solves 8 additional contigs with 14 extra fragments: this results in 51/59 contigs ending up with a copy number estimation. The 8 unsolved contigs are explained by the low quality of the coverage information that they have: very few points with a high degree of depth variation (contigs 41, 51, 52, 54, 53 and 58) or no coverage information at all (contigs 46 and 48 – supplemental material 72).

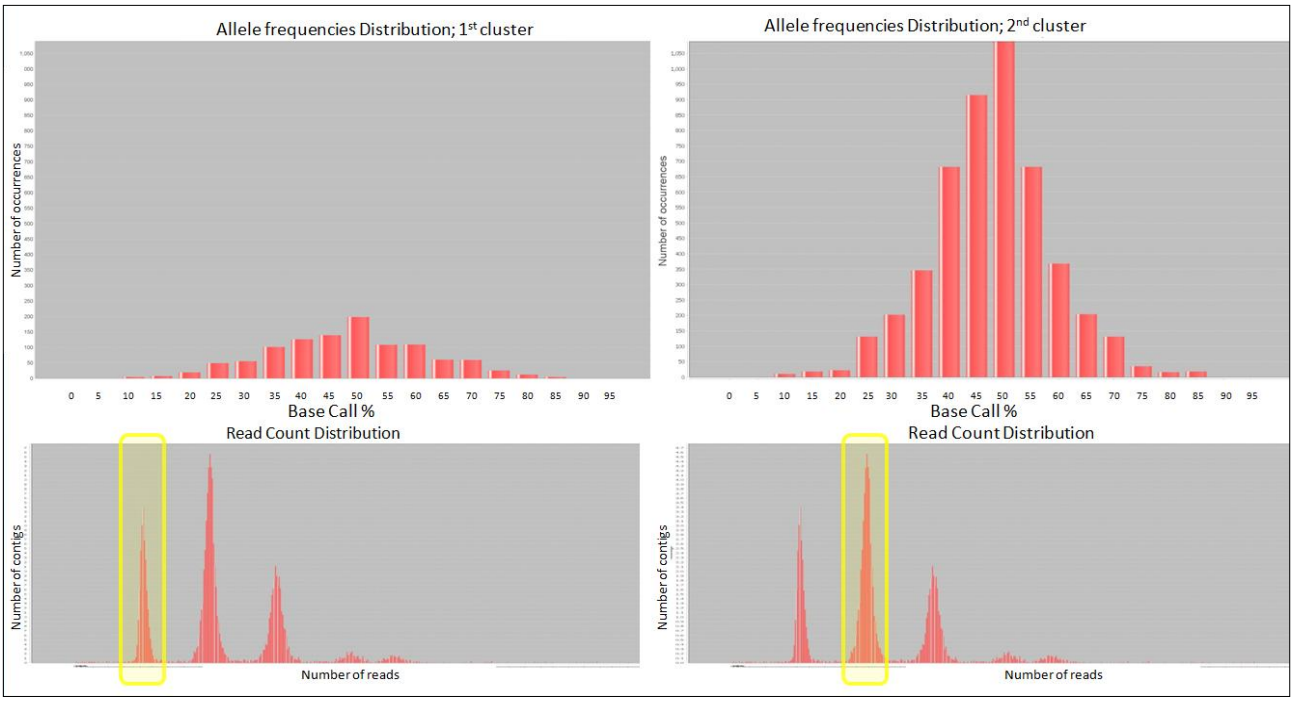


Figure 20 Allele frequency analysis (top figures) for Baseclear CBS1483 shows the base call percentages of the variations found in reads from the first and second clusters of the read count distribution (bottom). Here we have a good example of the utility of plotting the second cluster, since the first plot (bottom left) corresponds to p_1=1, the shape is not distinct enough to be recognizable and could easily be mistaken for p_1=2 except that it has considerably less variations. The second cluster (bottom right) can never have p_2=1, and it is more clearly shaped around the 50% bin, indicating a p_2=2. Results are similar on Novogene CBS1483.

With *wl* ≥ 3.500bp, what seemed to be a long tail in the third cluster, starts to reveal two small clusters corresponding to a small percentage of the reads with higher copy number regions (**Figure 19 bottom**). The ploidies now reported are P={1, 2, 3, 4, 5}. There are two exceptions to this that happen with very large window sizes. In the first exception, the bias in the deviation of the clusters (due the sum of the Gaussians effect described previously) provides a report of ploidies P={2, 4, 6, 9} with *wl*=20 Kbp. In the second exception, an extremely big size of *wl*=75 Kbp also results in a very irregular read count distribution that recognizes false peaks, resulting in a report of P={2, 4, 5, 6}. In both cases, the basic ploidy is twice as big as the one reported in the rest of the estimations. When this happens, the error score of the fitting is significantly higher, which triggers the allele frequencies analysis.

With the mentioned window lengths, PEDCA runs an analysis on the contigs having continuous ploidy estimation equal to the two first clusters in the read count distribution. For instance, if the first

cluster corresponded to an estimation of p_1=2, and the second one to p_2=4, we'll measure the variation of all contigs having a unique contiguous ploidy = 2 or 4, because the reads that map to those contigs are the ones that form the first two clusters. In the case of CBS483, the allele frequencies plot determines that the second cluster has $p_2 = 2$ and therefore the first one $p_1 = 1$, which validates the most reported estimation P={1, 2, 3, 4, 5} (**Figure 20**)

We run PEDCA with a large range of different window sizes and find that the peaks were correctly fitted and recognized even with lengths as large as *wl*=75 Kbp (supplemental material **70**). The *wl*=6.0 Kbp detects a false peak in the read count distribution that prevents PEDCA from finding a logical ratio explanation. When this happens, PEDCA stops running and issues a message in the main output file to warn the user about the situation and prompts him to run the program with a different window size.

The current implementation runs a first estimation with an initial *wl* then runs a second one with a new size adapted to the smallest contig. This feature helps to estimate eight extra contigs on the Baseclear sequencing. As previously explained, the first round with a large *wl* helps sampling the read count distribution with more smoothed values. This results in more distinct clusters and a higher reliability of the inferred PURC. A second round solves the remaining contigs with a smaller *wl*.

Yet, there is another limitation to the size of the first window length. When a contig is not sampled with enough coverage points in the first round, it is left to be solved in the second one. The second window length is adapted to the smallest contig; thus, when very long contigs are left, a short sampling bin will provide far too many points with too much standard deviation in the coverage cloud which leads to a very fragmented and less reliable copy number estimation. This is particularly the case when the contig lengths of the reference sequence have a big standard deviation and PEDCA has to deal with both very large and very small contigs at the same time. The solution to this problem is to provide a window size for the first run adapted to as many large contigs as possible. There is a range of values for optimizing the fragmentation problem.

Figure 21 shows the fragmentation in the first and second rounds with a range of initial window lengths going from very small (50bp) to very large (75 Kbp). The details of the plot are also available in a table format in the supplemental section (**Supplemental Table 4**). The less fragmented estimations are obtained with sampling bins in the range of 400 bp to 3.500 bp. With original *wl*'s < 400bp, the breaks in the contigs of the first round are higher than those in the second. The few contigs that are left for a second round have a small size and can be easily be estimated with the small *wl*. But as the original *wl* is set higher, and especially above 5.000 bp, bigger contigs cannot be solved in the first round. When sampled with the second *wl*, (which is too small for them), their coverage variation is too high leading to very fragmented and unreliable ploidy estimations. As the original *wl* increases, so does the second run number of breaks. The plot shows that the region where the breaks from the first and the second round cross each other corresponds to the range of ideal window sizes that lead to the less fragmented estimations. The final copy number inference for the Baseclear sequencing will combine estimations made with *wl*=400, 500, 750 and 1000 bp (supplemental material **73**)

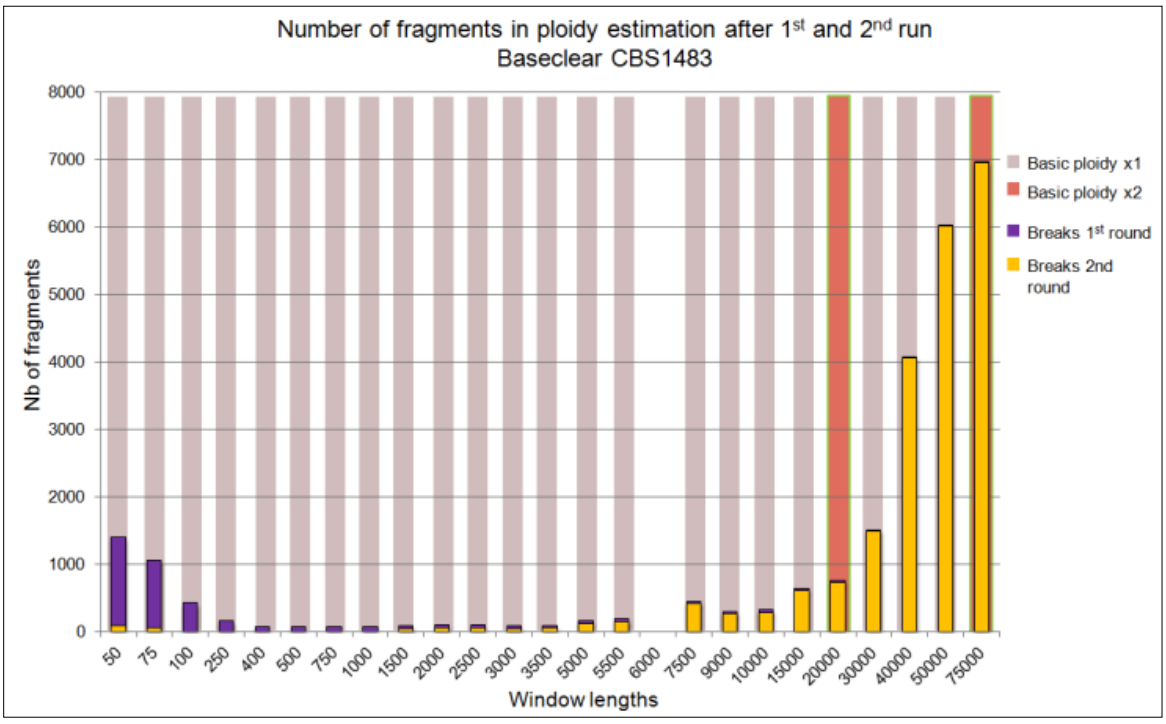


Figure 21 Ploidy estimation fragmentation by window length for Baseclear CBS1483. The figure shows that the optimal size range for getting the most continuous estimations are between *wl*=400 bp and *wl*=3.500 bp. Ploest could not run with *wl*=6Kbp, and two bins estimated a ploidy twice as big as all the others (wl= 20 Kbp and 75 Kbp)

Results on the S.pastorianus CBS1483 Novogene

Overall, the results on the Novogene sequencing don't differ much from the Baseclear, which is to be expected as it is the same organism in both sequencings. However, it is a surprisingly good result given that this dataset was not able to run using previous methods (Magnolya) with more clear estimations than its Baseclear equivalent.

Results from Novogene show ploidy estimation for up to 52/59 contigs, one more than the Baseclear version. Curiously, a ploidy of p=3 is estimated for Scaffold 55 (741 bp) whose data wasn't continuous enough in the Baseclear sequencing to allow a copy number output.

By looking at its fragmentation per window length plot (**Figure 22**), we can see that the data set is more prone to cluster detection errors, failing to infer a proper ratio with *wl*= 75, 100 and 20.000 bp. As for the Baseclear sequencing, the copy numbers more often reported are P={1, 2, 3, 4, 5}, (where $p_1 = 1$ and $p_2 = 2$) but with the Novogene set of reads, we have a more frequent report of a basic ploidy twice as big ($p_1 = 2$ and $p_2 = 4$), each time with large window sizes (9, 30, 40, 50 and 75 Kbp). This only occurred twice with the Baseclear dataset. The allele frequency analysis could disambiguate the ploidy with the same result as in the Baseclear sequencing, giving similar plots to those shown in **Figure 20**.

We obtain the same plot showing more fragmented reports with small and big window sizes, but the optimal range is a bit larger than in the Baseclear sequencing. This means a higher resistance to extreme window lengths (especially the shortest ones) in terms of fragmentation, which is a first sign suggesting that the Novogene data has less standard variation than the Baseclear.

A comparison of the coverage and ploidy plots between the two datasets (supplemental material **80**) confirms this analysis, which is quite surprising. Due to the incompatibility of Novogene CBS1483 with previous methods, we expected it to be a noisier dataset with more standard variation than its Baseclear counterpart. Instead, the Novogene data set is more compact and clean, which also

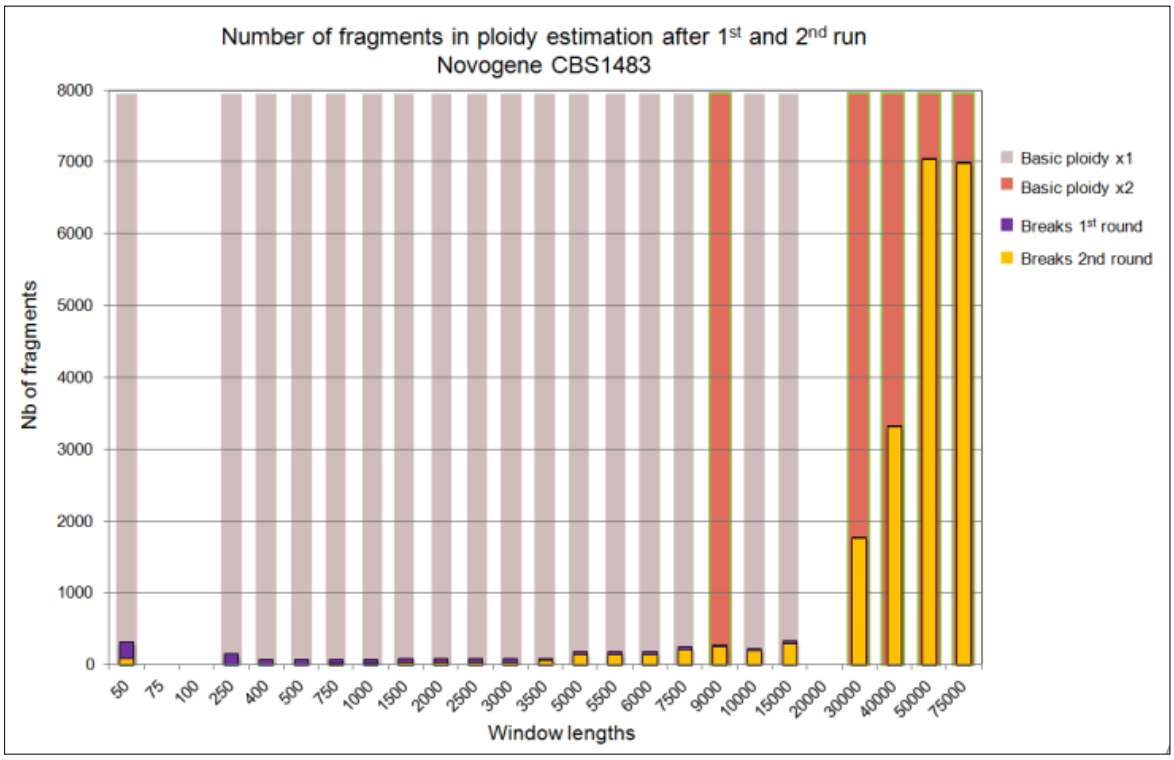


Figure 22 Ploidy estimation fragmentation by window length for Novogene CBS1483. Best sizes for getting the most continuous estimations are similar to those for Baseclear. PEDCA could not run with *wl*=75bp, 100 bp and 20 Kbp, and five bins estimated a ploidy twice as big as all the others (9, 30, 40, 50 and75 Kbp).

explains why it is possible to infer the copy number of Contig 55 with this data set (as well as other small extra fragments from different scaffolds).

If we output a selection of the less fragmented estimations with different sampling sizes and we compare it with the Baseclear estimations, we notice some minor differences. Some additional fragments that were not estimated previously are identified here. A small sequence in Contig 35, from position 224 Kbp to 229.5 Kbp (+/- 1.225bp) is estimated with *p*=4 where no ploidy was previously estimated. Another unidentified segment at the end of contig 43 is identified with *p*= 5. Contig 50 displays constant *p*=1 with Novogene, but a dubious small change is detected with Baseclear of *p*=2. This could also be explained by the difference between both data sets' standard variations.

Much more spectacular and clear is a divergence in Contig 24. This scaffold is 652 Kbp long and its size excludes the possibility of such a difference being explained by standard variation. While the Baseclear genome has a clear *p*=1, the Novogene outputs a solid estimation of *p*=2 over the whole extent of the scaffold. This intriguing scaffold maps to Chromosome X of *S.Pastorianus* and seems to correspond to a recent mutation in the strain sequenced by Novogene (**Figure 23**).

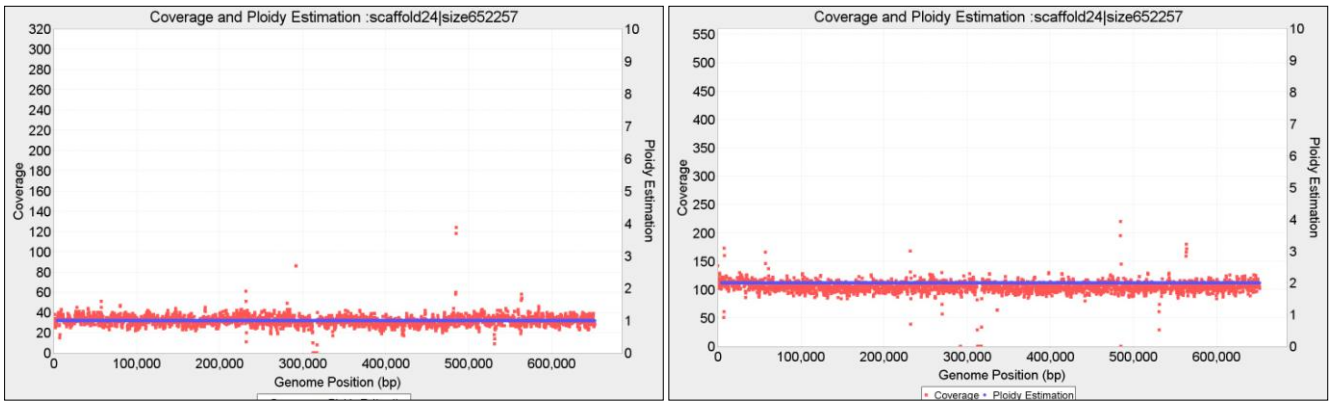


Figure 23 Baseclear (left) and Novogene (right) CBS1483 reads aligned to the scaffold 24 (652 Kbp) clearly displaying a different ploidy of 1 and 2 respectively. It is the only whole contig for which the two sequencings differ.

We also obtained a 2017 actualized assembly version of CBS1483 from the Delft's Department of Biotechnology that corresponds to the estimation obtained by Van den Broek in 2015. This version has integrated it own mapping to the ancestors' making it easier to compare our chromosome structures against the 2015 results.

Using the 2017 reference yielded identical estimations by PEDCA when using the Baseclear or the Novogene library, except for chromosome 'Seub10-Sc10' which confirms the difference already reported in 'scaffold 24' (**Figure 23)** with the 2015 reference. Since our Baseclear library is the same one used by Van den Broek, we can compare the results with both programs when mapped on the two references. **Table 1** displays comparative results between PEDCA and Magnolya.

We refer the primary ploidy to the largest fragment estimated with a given copy number on a contig. The second largest fragment with different copy number will be the secondary ploidy and so on. A small secondary ploidy corresponding to a structural variation in 'Seub6' is detected by PEDCA but not by Magnolya. PEDCA also detects a secondary ploidy in 'Sc15Sc11' that appears to be detected by Magnolya but not reported in the final visual representation of the chromosome structures in the 2015 paper.

Inversely, Magnolya reports 3 secondary ploidies that are not detected by PEDCA: 'Sc10-Seub10' $p_2 = 1$; 'Seub1' $p_2 = 1$; and 'Seub12' $p_2 = 4$. Magnolya's estimation of 'Seub10-Sc10' corresponds with PEDCA only with the Baseclear sequencing.

Results on the Trypanosoma cruzi CL Brenger genome

Trypanosoma cruzi, a triatomine parasite (also known as a "kissing bug") is the infecting agent of the commonly known Chagas disease. Symptoms of the infection can include fever, flu-like symptoms, a rash or swollen eyelid. Early symptoms usually go away, but the infection can cause serious intestinal and heart problems if left untreated, (Ley, Andrews, Robbins, & Nussenzweig, n.d.)

The American Center for Disease Control considers it as one of the five neglected parasitic infections and, as such it has been targeted for public health action **(CDC, 2011)**.

T.cruzi's genome is not well known. One strain, the *T.cruzi* CL (TcVI lineage), is also the result of a hybridation of two related ancestors: the TcII and TcIII lineages. A Sanger reference genome was published in 2005, generated by whole genome shotgun with Sanger reads (El-Sayed N. et al., 2005).

The genome has a high repetitive content of about 50% of its size. Its ploidy is currently unknown and about 1/3 of the assembly consists of unassigned contigs (~36 Mbp) that have yet to be mapped to any of the ancestors' haplotypes. It is an extremely noisy data set on which we can test the limits of our method.

We first map Illumina paired end reads (2 x 150bp) to the respective 40 pseudo chromosomes references of the two ancestors known as Esmo-like (TcII) ~35 Mb and Non-Esmo like (TcIII) ~32.5M bp scaffolded by (Genomics, Weatherly, Boehlke, & Tarleton, n.d.) in 2009. The highly repetitive content makes the ploidy estimation very difficult, and even if we can identify some areas with defined and continuous coverage, the results remain very fragmented and ambiguous (**Figure 24**).

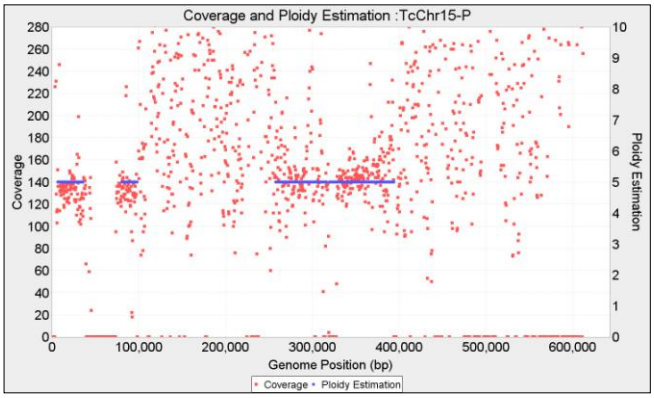


Figure 24 *T.cruzi* Illumina library mapped to TcIII-like (Chromosome 15) *wl*=750 bp. While some areas contain coverage information clear enough to allow ploidy estimation, big segments with very high coverage variability due to repeated sequences, make the estimation very unreliable. Larger window sizes are less sensitive to repeats because small repetitive regions are more frequent than big ones, but this genome remains quite noisy even with large bins.

CONTIG	2017 Pedca P1	2017 Pedca P2	2015 Mag P1	2015 Mag P2
Sc01	3	4	3	4
Sc2	3		3	
Sc3	-	-	-	-
Sc4	3		3	
Sc5	3		3	
Sc6	1		1	
Sc7	1		1	
Sc8	5		5	
Sc9	3		3	
Sc10-Seub10	2		2	1
Sc11	2		2	
Sc12_sf1	2		2	
Sc12_sf2	2		2	
Sc12_sf3	2		2	
Sc13	2		2	
Sc14	2	3	2	3
Sc15Sc11	3	1	3	
Sc16	2		2	
Seub1	2		2	1
Seub2-04	1		1	
Seub03-Sc03	4	3	4	3
Seub04-02	1		2	
Seub5	2		2	
Seub6	3	4	3	
Seub07-Sc07	3	4	3	4
Seub08-15	1		1	
Seub09	2		2	
Seub10-Sc10	Bc:2 Ng:1		1	2
Seub11	2		2	
Seub12	2		2	4
Seub13-Sc13	2	4	2	4
Seub14-sc1	2		2	
Seub14-sc2	2		2	
Seub14-sc3	2		2	
Seub15-08	1		1	
Seub16	2	4	2	4

Table 1 Chromosome structures obtained mapping sequenced reads from Baseclear to different references of S.pastorianus. First two columns display the results using PEDCA against a 2017 reference of CBS1483. 'Pedca P1' is the main ploidy reported and 'Pedca P2' the secondary. The two last columns are the results reported by (van den Broek et al., 2015) using Magnolya and the Baseclear libraries with 'Mag P1' being the main ploidy and 'Mag P2' the secondary. PEDCA results are identical with Baseclear (Bc) and Novogene (Ng) sequencings except for chromosome Seub10-Sc10. Magnolya results are based on Baseclear reads only. Differences between Magnolya and PEDCA are highlighted in yellow.

The read count distribution of the separate ancestors is very similar. Both have two similar clusters that were commonly identified as ploidy 2 and 3 (**Figure 26 top**).

We also map the same Illumina reads to the BroadTcVIPacBio reference. The assembly consists of a very large number of contigs (2.697).The mean length is 32 Kbp with a contig N50 of 83.077 bp and a N90 of 12.236 bp. We decided to run our tests only on the 50 longest contigs.

The BroadTcVIPacBio reference allows the identification of a third cluster corresponding to p_1=1 (**Figure 26 bottom**). It is interesting that this ploidy don't appear when mapping the data to any of the ancestors. The estimations per contig confirm this result. While most of the scaffolds are diploid, a considerable number seem to be triploid and some of them are haploid. The ploidy estimation that results from using this reference is significantly better than from using the ancestor's genomes. The estimations are much less fragmented and longer regions are estimated, which reflects a closer proximity of the real genome to the BroadTcPacBio assembly than to the mere sum of its two ancestors. (supplemental material **88**)

It is worth underlying the utility of the adaptable widow size in the analysis of the different contigs which allows comparing results from different sampling sizes. Contig 16's coverage in the BroadTcVIPacBio alignment shows higher variability than its fellow contigs and with *wl*=1000 bp it is still difficult to determine the overall ploidy (**Figure 25**). By switching to a higher *wl*=7000 bp, with data points averaged over a longer length, the estimation becomes more clearly defined as *p*=3.

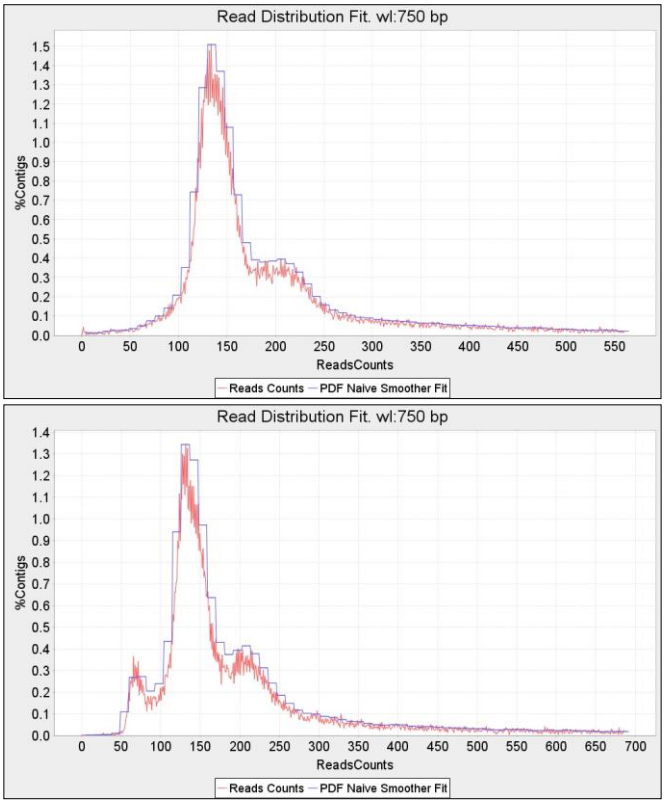


Figure 26 Read count distribution for T. cruzi. Similar to the TcIII strain, the Esmo-like alignment (top figure) results in two clusters that are usually recognized as ploidies 2 and 3. The same reads aligned to the BroadTcVIPacBio assembly (bottom figure) reveal a third cluster corresponding to *p*=1.

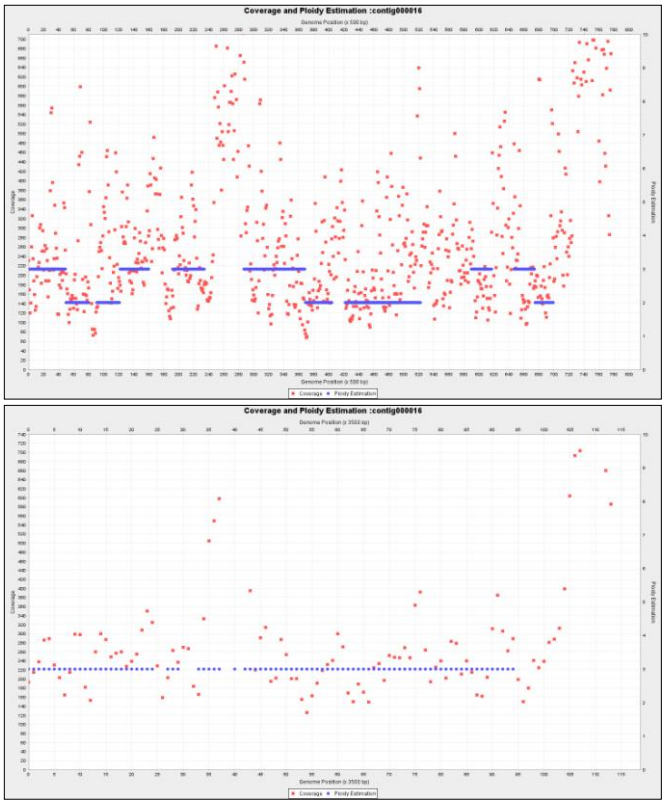


Figure 25 Running the algorithm with a range of different window lengths is useful for tuning the ploidy estimation. The analysis of Contig 16 of BroadTcVIPacBio with *wl*=1.000 bp (top) is more ambiguous than with *wl*= 7.000 bp (bottom)

Table 3 displays PEDCA results on *T.cruzi* run with different window sizes (2, 3, 5 and 7Kbp) and selecting the neatest estimations. Overall the estimations are relatively continuous; this is partially due to the use of long window sizes. Contigs 5, 15, 26, 43 and 44 have important areas that remain unsolved, but usually less than half of the length of the contig.

Often, some contigs (6, 9, 17, 19, 23, 26, 35 and 41) have fragmented estimations that are less clearly defined, making it difficult to decide whether the breaks are due to a real ploidy change or just a fluctuation in the coverage (**Figure 25 top**). Incrementing the window size sometimes helps to disambiguate these situations, but not always. Some cases are very difficult to solve when the data is dispersed along the area between two ploidy values, and it is impossible to determine whether the coverage reflects under-mapping or over-mapping of reads to that region.

In other cases, the estimation is not fragmented but raises doubts as to whether the continuity is due to a real unique copy number or simply the result of averaging over a long window length (Contigs 3, 14, 16, 18, 20, and 39). This raises a legitimate warning against using very long window sizes that could result in very few averaged points with low reliability of the estimation, especially with highly dispersed coverage data (**Figure 25 bottom**).

Contig 48 remains completely unsolved with discontinuous coverage values scattered across the extent of the sequence.

Still, we have supported full length estimations for 30/50 contigs plus partial or draft estimations for some additional 19 sequences, which is a relatively good result for this particularly noisy genome with highly repetitive content.

PEDCA's time performance

We run PEDCA in a personal laptop Intel(R) Core(TM) i5-4200M CPU 2.50 GHz with 8.00 GB of RAM. We run tests on our simulated dataset, the two *S.Pastorianus* sequencings (Baseclear and Novogene CBS1483) and in *T.cruzi* strains II and VI. We measure PEDCA performance with different *wl*'s. We also compare the times with and without the allele frequencies analysis (**Table 2**)

Assembly	bam (Gb)	TIME WITH bam FILE ONLY (secs)					vcf (GB)	TIME WITH bam FILE + vcf FILE (secs)				
		wl=500	wl=750	wl=1000	wl=2000	wl=3000		wl=500	wl=750	wl=1000	wl=2000	wl=3000
Simulated	3.12	67	58	60	56	56	0.52	126	115	103	95	99
S.pastorianus												
CBS1483Baseclear	1.17	33	29	26	26	25	3.07	250	172	187	206	198
CBS1483Novogene	1.66	58	58	46	53	51	3.13	249	227	239	224	227
T.cruzi												
Illumina_TcII	6.71	140	128	132	146	134	4.22	5064	6777	5640	6514	5892
PacBioBrenerTcVI	6.77	136	131	148	174	126	2.66	271	272	282	287	291

Table 2 PEDCA's time performance (in seconds) for the simulated data set, both CBS1483 sequencings (Baseclear and Novogene) and *T.cruzi* II and VI. When only the .bam file is provided, the size of the file affects the speed of the algorithm but it generally runs quite quickly (between 25'' and 174''). When an allele frequency analysis is requested, the variant call parser seems to be quite sensitive to the length of the .vcf file, having runtimes of almost up to two hours in the case of T.cruzi II.

Overall results show that PEDCA performs quite quickly when only the default analysis is performed. Even with the largest .bam files (*T.cruzi*, 6.7 Gb), run times remain under three minutes. Run times remain under a minute with smaller files such as those from both versions of *S.Pastorianus*.

The allele frequencies analysis generally adds two or three extra minutes of processing, but something strange occurs with the TcII assembly which suddenly extends the analysis of the ploidy to 1½ hours (sometimes almost 2 hours). The slight increment in the size of TcII .vcf file (4.22 Gb) in comparison to the rest of the assemblies (0.52, 2.66, 3.07 and 3.13 Gb) doesn't justify such a difference. The algorithm to perform allele frequency is quite straightforward and consists of a .vcf parser and a graphic interface to plot the histogram, which makes the time complexity of the process linear. We could not find a reasonable explanation for this behavior, but it might be related to the very highly repetitive content of the genome.

Ploidy discussion, limits and perspectives

We've developed PEDCA, a method that runs fast ploidy estimations and is easily exportable to any operating system with a Java platform. It only requires a .bam or .sam file as input and provides an optional supplementary analysis if a .vcf file is provided. PEDCA output depends mainly on one single parameter: the window length. This makes it easy to operate while allowing for fine tuning with a set of secondary parameters for more precise usage.

We've built a sophisticated simulated genome from real sequenced data to obtain a reference with characteristics as similar as possible to the real *S.Pastorianus*. We've provided a solid proof of concept validation of PEDCA by testing it in our simulated dataset. Our results on real data show that the method works by comparing it with previously solved data (Baseclear CBS1483) and also works with datasets that previous methods could not solve (Novogene CBS1483).

The different features that characterize PEDCA were essential to identify ploidy differences (a few small and one quite large) amongst the two versions of the CBS1483 strain. This analysis reflects that genome copy number in this particular brewing yeast is highly dynamic.

We've also extended this study to obtain the previously unknown ploidy of *Trypanosoma cruzi* TcVI (a very noisy genome with highly repetitive content) that is very different from *S.Pastorianus*; and we've proven that, with only some small tuning in the window length, the method responds well to a significantly different genomic profile.

We've proven the utility of the dynamic window length feature by implementing a second round of coverage sampling. We've also provided evidence of the utility of incorporating an allele frequencies plot to facilitate analysis of the PURC. This feature is run by default if a .vcf file is provided and can help disambiguate the basic ploidy estimations amongst all of the potential solutions.

But PEDCA's performance is affected by the following factors:

- PEDCA estimation relies on the alignment of a set of reads to a given reference. Low quality references and/or bad alignment of the reads might result in chaotic or erroneous coverage data sets with incorrect ploidy estimations.

- High repetitive content, as we've seen, leads to over-reported coverage, especially when the reads length is shorter than the repetitive regions. The alignment tool might also fail to correctly map reads from a different haplotype when too much heterozygosity is present. This can lead to under-reported coverage in some segments. Generally speaking, any factor affecting the continuity of the coverage makes the ploidy harder to evaluate.

- Another limit is that high ploidy becomes harder to identify. In some contigs of CBS1483, we were able to detect segments with copy numbers of eight. We estimate that the ploidies that PEDCA can detect are those below ten, but the reliability of the analysis depends on the quality of the coverage information and its variation.

We've tried to palliate these problems with different features. We tried to average values and insure continuity of coverage with diverse parameters before estimating the copy number of a contig; but ultimately, high standard variation and discontinuity of the data are two major drawbacks in the depth of coverage analysis.

There are a few ways in which PEDCA could be improved in the future.

Among the possible improvements that could be implemented, one that follows the natural evolution of PEDCA would be a method that increases the flexibility of the window length and automatically customizes it to each contig size. In the current version of the tool, the *wl* only changes between the first and the second run; finding the optimal length per contig could significantly increase the quality and reliability of the estimations.

Sometimes a *wl* is on the threshold of allowing a contig to be solved on the 1st run. When long contigs are just a little too small to be solved in the first run and the window is adjusted to solve the shorter ones in the second run, the estimation of the longest contigs that make it to the second run have too much data and variation to be solved properly. This would also be solved by a more adaptable window.

For contigs smaller than 1000 bp, having a sliding window might not be so useful. In such cases, it is probably a good idea to implement a feature that averages the coverage over the whole contig when this happens. The irregularities over the sequence would be smoothed; it would maximize the probability of having enough data to provide a estimation; and the risk of losing fragments with changes in ploidy should be acceptable for such a small size.

A measurement is needed to score each estimation in order to compare results from different window lengths. This measurement has to take into account different elements: the standard deviation of the coverage cloud around each different ploidy estimation; the different estimations within a single contig (fragmented ploidies); and the proportion of the contig that is estimated in relation to the total length of the sequence. The score should also somehow take the density of the data points/sampling into consideration. With such a measurement, it would be easy to compare the output from different window lengths for the same contig and choose the best result. The algorithm could then run without requiring any other input beyond the .bam file, simply by running a spectrum of adapted or preselected window lengths and choosing the best customized ploidy estimation for each contig among all of the available contigs (see example of **Figure 25**) . This would be an alternative to the customized *wl* and could be relatively easy to implement in the next version of PEDCA.

Another useful feature to implement in a subsequent version would be an automatic recognition of the frequency allele distribution by recognizing the specific shapes proper to each copy number. The feature could launch a second run automatically when it detects a wrong coverage to ploidy ratio. The second run would force the PURC inferred by the allele frequencies analysis.

CONTIG	Length	wl	Ploidy 1	from	to	Ploidy 2	from	to	Ploidy 3	from	to
1	1254	2000	2	0	125.4						
2	849	5000	3	0	849.4						
3	844	5000	3	0	844.2						
4	682	2000	2	0	940.0	1	940.0	103.2	2	103.20	115.30
5	643	2000	1	0	435.0	3	435.0	508.0			
6	562	3000	1	0	330.0	2	330.0	639.0			
7	494	2000	1	0	494.5						
8	482	2000	1	0	482.8						
9	453	2000	1	0	453.8						
10	445	2000	1	0	120.0	1	143.0	415.0			
11	444	2000	1	0	444.7						
12	438	2000	1	0	438.6						
13	435	2000	1	54.0	405.0	2	0	54.0			
14	429	2000	1	144.0	574.0	2	66.0	144.0	1	0	66.0
15	417	2000	1	0	325.0	2	325.0	470.0			
16	402	7000	3	0	402.8						
17	393	5000	2	0	393.9						
18	384	5000	3	0	384.9						
19	382	5000	2	0	135.0	3	135.0	370.0	2		
20	375	5000	2	0	167.5	2	252.5	362.5			
21	370	5000	3	0	370.7						
22	364	5000	2	0	364.6						
23	359	5000	2	290.0	495.0	1	0	290.0			
24	346	5000	2	0	347.0						
25	339	5000	2	0	339.0						
26	326	5000	2	0	345.0	3	345.0	445.0			
27	318	5000	2	0	318.2						
28	308	5000	2	0	309.0						
29	306	5000	2	0	306.7						
30	303	5000	2	0	303.8						
31	302	5000	2	0	302.9						
32	301	5000	2	0	301.3						
33	300	5000	2	0	300.8						
34	300	5000	2	0	300.1						
35	288	5000	2	0	290.0	1	290.0	427.5			
36	282	5000	3	0	282.7						
37	281	5000	2	0	281.4						
38	273	5000	2	0	273.7						
39	271	5000	3	0	271.3						
40	270	5000	2	0	270.7						
41	267	5000	1	190.0	317.5	2	0	190.0			
42	267	5000	2	67500	267.0						
43	266	5000	3	0	266.6						
44	265	5000	2	0	265.9						
45	263	5000	2	0	263.1						
46	261	5000	3	0	261.8						
47	260	5000	1	0	260.1						
48	258	5000	-								
49	257	5000	3	0	257.1						
50	254	5000	2	0	254.1						

Table 3 Ploidy estimations using PEDCA on Illumina reads from *Trypanosoma cruzi* VI mapped over the 50 longer contigs of the BroadTcVIPacBio assembly. 4 different window lengths (*wl*) were used: 2.000, 3.000, 5.000 and 7.000 bp. Three possible copy numbers estimations (order by length) are shown in columns 'Ploidy 1', 'Ploidy 2' and 'Ploidy 3'. Only 10 contigs have fragments with two different ploidies, and only two of them have three different ploidy fragments. The columns 'from' and 'to' indicate the beginning and end of the fragment with such a copy number.

Phasing

Intro to phasing

We created a new depth of coverage algorithm to estimate the copy number of contigs, but inferring chromosome copy numbers alone doesn't provide information on the variations along the draft reference genome. Two further steps are required in order to obtain the full detailed sequence of each parallel sequence: detecting the variations in our reads when aligned to our consensus reference and phasing them.

We can detect the differences between the haplotypes with a variant caller like Pilon (Walker et al., 2014) which detects heterogenic variance along the draft reference genome, but having the list of variations relative to a draft consensus genome doesn't describe the detailed haplotypes of each chromosome copy.

In order to determine which combination of variations describes a haplotype, we need to assign each variation to the allele to which it belongs. For that, we need to have sufficiently large reads to span the longest distance that separates variation positions, and then cluster those that are consistently found in the same copy. The scope of this research is limited to solutions that work with short reads. Paired end reads provide an extra advantage to the problem since they have insert sizes that partially palliate the short length of the reads; however, the insert sizes have relatively fixed lengths. They have some standard variation that provides some, but not enough, flexibility to cope with the unpredictability of the possible distance among variation positions that we need to solve.

Here we describe some steps taken towards a new approach on polyploid phasing that tries to group SNP's together in their respective strain by cleaning the variant information obtained from the alignment and discriminating sequencing and alignment errors from real allele variants.

Haplotyping methods

Here we describe some steps taken towards a new approach on polyploid phasing that optimizes the variant information obtained from the alignment by grouping SNP's together in their respective haplotype.

In this paper, we define a haplotype as the nucleotide sequence along a single copy of a chromosome. Because the chromosome might have more than just one copy, each haplotype will have a very similar sequence of nucleotides, but with its own specific variations at certain loci.

Phasing or haplotyping is the process of clustering together the variations that belong to the same haplotype, which defines the correct nucleotide sequence for each chromosome copy.

(Motazedi, Finkers, Maliepaard, & de Ridder, 2016) evaluated the performance of three state-of-the-art haplotype estimation algorithms for polyploids: HapCompass (Aguiar & Istrail, 2012), HapTree (Berger, Yorukoglu, Peng, & Berger, 2014) and SDhaP (Das et al., 2015), with different levels of sequencing depth, ploidy levels and genomic diversity, using tetraploid potato as the model. They conclude that sequencing depth is the major determinant of phasing reconstruction quality and that

1kb PacBio CCS reads and Illumina reads with large insert-sizes are competitive; however all of the exiting methods fail to produce good haplotypes with increase in ploidy.

They also signal a particular challenge of haplotyping polyploid genomes. Diploid phasing is different from polyploids because the alternative variant in diploids is always present in the other allele, which eliminates the need to calculate possible combinations. However, having to compute the combinations of the potential positions of the different alleles leads to a computational problem of exponential complexity when the ploidy increases.

When representing different haplotypes of a same chromosome, we can ignore the homozygous regions and only consider the variant sites. For a chromosome with *k* variants, we can represent its haplotype as a string from the set {'A','C','G','T'}. Even though the possibility of tetra-allelic loci exists, it is often assumed that variants are bi-allelic. We accept multi-allelic sites in our method.

The objective behind our approach is to tackle the polyploid haplotyping problem in order to avoid having to compute all possible combinations of SNP's as that is computationally very expensive. Most methods don't avoid the exponential approach. Instead, they find ways to reduce the computational complexity with coding strategies, mostly with dynamic programming.

The exponential problem arises when, in order to separate the reads with errors from those which really describe the different haplotypes, all possible variant combinations are computed and then the likelihood of each combination is computed, with the most likely combinations being retained and the others discarded.

Here, we explore the possibility of avoiding computing all combinations by removing the error variations before clustering the remaining ones in the correct number of haplotypes. Thanks to PEDCA, we have the advantage that we can infer the exact number of copies per contig that we are looking for beforehand. The idea is to only consider the error-free variants that are supported with enough coverage and to cluster them in the number of haplotypes given by PEDCA.

Our exploration uses as input a .sam/.bam file with all of the reads aligned to the reference genome, and the .vcf files with all the variants of the reads aligned to their references. The .vcf file reports all variations relative to the provided reference found in the available libraries. The reads might contain sequencing and alignment errors; our objective is to discard both of them.

The basis of our approach is to first identify all positions where variations are detected. A .vcf file stores the percentages of different alleles present per position with respect to a draft reference. Because we know the expected number of haplotypes *p* per contig, we can discard the variations that don't reach a certain safe threshold of presence 1/2*p* (half of the contig copy proportion 1/*p*). This means that the remaining detected variations have enough coverage to represent at least one copy of the reference. We end up with a list of positions where enough variations are detected to discard errors. We also keep track of the actual SNP's that can be observed (**Figure 27**). We can define an observation by the pairing of a position (or loci) *l* with the variation *v* observed at that position. We can then represent all of the observations as a vector of pairs

$$O = \{l_1|v_1,\ l_2|v_2,\ l_3|v_3 \ldots\ l_n|v_n\}.$$

The sequence of the observations is composed of a string from the set that includes the four bases and the deletion {‘A’,’C’,’G’,’T’,’-‘}. In our toy example (**Figure 27**), we observe three alleles at position 16: ‘AACC’, ‘A-‘ (‘A’ followed by a deletion of AAC in respect to the padded sequence), and ‘G’. Three observations will be considered and stored as ‘16AACC’,’16A-‘and ‘16G’.

Since we have different libraries of paired ends with diverse insert sizes, the possibility of connections between distant variation loci is bigger, but not big enough to insure coverage for all possible combinations. Our situation becomes an optimization problem.

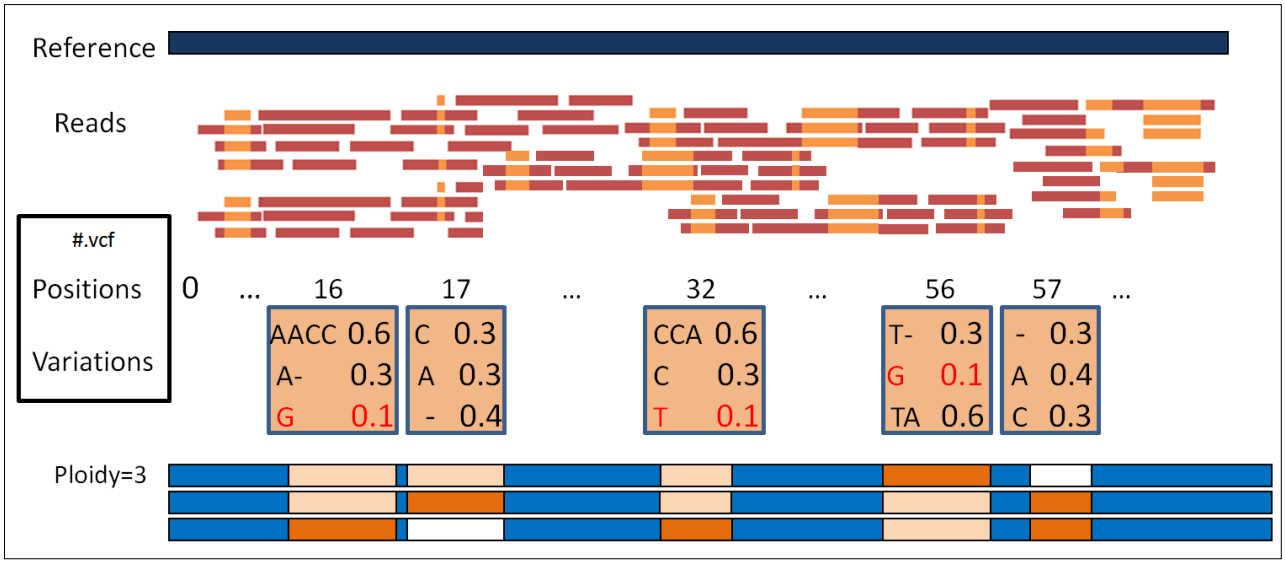


Figure 27 When the reads are mapped to the reference, the variations, their rates and their positions are stored in a .vcf file. Beforehand, we had estimated the ploidy for each contig, (p=3 in this toy example) so that we can reject the variations rate that are below a certain threshold (here = 1/2p = 0.165). In this example, position 16 has three possible alleles: ‘AACC’, ‘A-‘(‘A’ followed by a deletion ‘-‘), and ‘G’. But the variation ‘G’ (in red) is represented 1/10 of the time, which is below the threshold and therefore considered as an error. Same goes for variation ‘T’ at position 32 and ‘G’ at position 56. With enough coverage, the retained variations reflect the proportions present in the haplotypes.

The next step is to store the information of the reads that support such variations at those particular positions. We want to know which reads support each of the possible variations at the concerned loci, but we are also interested in knowing all of the variations that are simultaneously covered by the same read. Except for exceptional hybrid errors, each read reflects a singular haplotype so co-observed variations belong to the same chromosome copy.

We create a matrix H of dimensions n x n with all observations $O_{i=1\to n}$ in the rows and $O_{j=1\to n}$ in the columns. We store the reads that support their simultaneous co-observation at each intersection H_{ij}.

The read being aligned to a reference may have inserts or deletions that are not present in the reference, making it difficult to identify the relative position of one towards the other. The CIGAR string is a sequence of base lengths and the associated description of the read alignment to the padded reference (H. Li et al., 2009). They carry information such as which bases align (either as a match or a mismatch), which bases are deleted, and/or which are insertions that are not in the reference. In order to precisely define the loci where an observation maps to the reference, we need to reverse the way in which the read was aligned, then compare the sequence at the precise loci and store it.

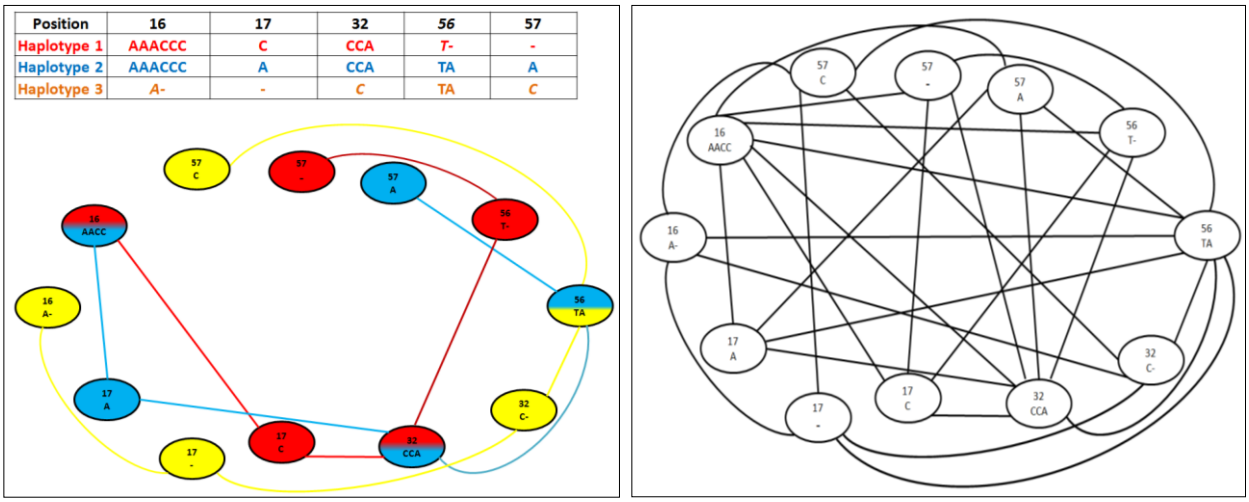
We use the CIGAR of the .bam file to fill the different intersections of the matrix. If an observation in a read doesn't match any of the identified paired loci/variation, that segment of the read is considered a sequencing or alignment error and is rejected. When a position in the read describes an alternative present in the matrix, the read has a match in the corresponding column of *H*. We then record the read identifier in every row of every other co-observed variation.

We can look at the result as an adjacency matrix of a graph of localized variations, in which the nodes describe the alleles and the edges, the co-observation of two variations in the same read. The number of reads recorded at each intersection H_{ij} is the weight that supports a connection between two variants. The intuition of the graph is described with a toy example in **Figure 28**.

The phased solution is a color graph with a number of different colors equal to the expected ploidy (we define each color by an integer). The challenge is to find a way to color the graph in an optimized way. There are a few particular properties of our graph that can help us to solve the problem.

First, some observations will be exclusive to one color/haplotype. This is certain for diploids and triploids; we also assume that there will always be some exclusive observations in higher polyploids if the sequences are long enough.

A second interesting property derives from the first one: if a variation is exclusive, then all the reads that traverse it belong to the same haplotype. Therefore, all the interjections with other positions in the matrix traversed by that read have at least that color. If we can identify those exclusive variations, we could have colored nodes that we can use as starting points to extend the colors along the rows and columns of the adjacency matrix.



Position	16	17	32	56	57
Haplotype 1	AAACCC	C	CCA	T-	-
Haplotype 2	AAACCC	A	CCA	TA	A
Haplotype 3	A-	-	C	TA	C

Figure 28 Assuming that we knew the correct observation sequence of each haplotype (top left table), we could represent the haplotypes as a colored graph with nodes representing the observed variations and the edges indicating existing connections between adjacent positions/variations (left graph). In reality, we do not have the color information but we do know which observed alleles are connected by examining the reads (right graph). Not only can we tell which connections exists among adjacent observations, but also among observations across the genome (since a read can span multiple variations) which results in a more complex graph. The challenge is to simplify the graph, extracting the connection information and coloring the nodes so that we reconstruct the left graph from the right graph.

	16AAACCC	16A-	17A	17C	17-	32CCA	32C	456TA	456T-
16AAACCC									
16A-									
17A	1 2 3 10 14 15 17 269140 269141 269148 149 150 155 156								
17C	4 5 6 11 12 13 16 18 142 145 146 151 152 157 158 159 160								
17-		7 8 9 143 144 147 153 154							
32CCA	1 2 3 4 5 6 10 11 12 13 14 15 16 17 18 140 141 142 145 146 148 149 150 151 152 155 156 157 158 159 160		1 2 3 10 14 15 17 140 141 148 149 150 155 156	4 5 6 11 12 13 16 18 142 145 146 151 152 157 158 159 160					
32C		7 8 9 143 144 147 153 154			7 8 9 143 144 147 153 154				
456TA	140 141 148 149 150 155 156	143 144 147 153 154	140 141 148 149 150 155 156		143 144 147 153 154	140 141 148 149 150 155 156 161 164 165 169	143 144 147 153 154 167 168		
456T-	142 145 146 151 152 157 158 159 160			142 145 146 151 152 157 158 159 160		142 145 146 151 152 157 158 159 160 162 163 166			

We start by identifying the observations in each read. Every read has its identifier. We fill the adjacency matrix with the read identifiers at the intersection of the co-observed variations.

	16AAACCC	**16A-**	17A	**17C**	17-	32CCA	32C	456TA	**456T-**
16AAACCC									
16A-									
17A	1 2 3 10 14 15 17 269140 269141 269148 149 150 155 156								
17C	**4 5 6 11 12 13 16 18 142 145 146 151 152 157 158 159 160**								
17-		7 8 9 143 144 147 153 154							
32CCA	1 2 3 4 5 6 10 11 12 13 14 15 16 17 18 140 141 142 145 146 148 149 150 151 152 155 156 157 158 159 160		1 2 3 10 14 15 17 140 141 148 149 150 155 156	**4 5 6 11 12 13 16 18 142 145 146 151 152 157 158 159 160**					
32C		7 8 9 143 144 147 153 154			7 8 9 143 144 147 153 154				
456TA	140 141 148 149 150 155 156	143 144 147 153 154	140 141 148 149 150 155 156		143 144 147 153 154	140 141 148 149 150 155 156 161 164 165 169	143 144 147 153 154 167 168		
456T-	142 145 146 151 152 157 158 159 160			**142 145 146 151 152 157 158 159 160**		142 145 146 151 152 157 158 159 160 162 163 166			

We then identify all of the exclusive variations and assign them each a color (starting in the toy example with '17C/red'). Since the observation is exclusive, all of the reads on it belong to the same color; we can therefore attribute a color to all the reads in that column and row.

	16AAACCC	**16A-**	17A	**17C**	17-	32CCA	32C	456TA	**456T-**
16AAACCC									
16A-									
17A	1 2 3 10 14 15 17 269140 269141 269148 149 150 155 156								
17C	**4 5 6 11 12 13 16 18 142 145 146 151 152 157 158 159 160**								
17-		7 8 9 143 144 147 153 154							
32CCA	1 2 3 **4 5 6** 10 **11 12 13** 14 15 **16** 17 **18** 140 141 **142 145 146** 148 149 150 **151 152** 155 156 **157 158 159 160**		1 2 3 10 14 15 17 140 141 148 149 150 155 156	**4 5 6 11 12 13 16 18 142 145 146 151 152 157 158 159 160**					
32C		7 8 9 143 144 147 153 154			7 8 9 143 144 147 153 154				
456TA	140 141 148 149 150 155 156	143 144 147 153 154	140 141 148 149 150 155 156		143 144 147 153 154	140 141 148 149 150 155 156 161 164 165 169	143 144 147 153 154 167 168		
456T-	**142 145 146 151 152 157 158 159 160**			**142 145 146 151 152 157 158 159 160**		**142 145 146 151 152 157 158 159 160** 162 163 166			

In the next step, we can extend the information of each colored read to all of its occurrences along the entire matrix, solving conflicts when they arise and coloring new whole columns and rows when new exclusive observations are found.

Figure 29

Third, the same principle of extending the color of exclusive observations through the matrix's rows and columns can be inverted. In the adjacency matrix, each read unique identifier is stored at the corresponding intersections. Once the color of one read has been defined, we can also extend its color to all of the intersections where its ID is found, which might in turn define new colors to new exclusive variations, and so on… **Figure 29**

The main problem is finding a reliable way to identify exclusive variations. A priori it could be possible to use the depth information to infer which observations have a number of reads proportional to the total coverage of their position which is equal to or sufficiently similar to 1/ploidy. Unfortunately, the coverage is not a constant parameter and its standard variation over the genome can easily overcome the minimum precision needed to identify exclusive variations, even with low copy numbers. Even when enough coverage is available, this approach can only work until the coverage variation becomes higher than our safe threshold. This approach becomes unreliable with higher ploidy because, as we've seen, the coverage standard variation increases proportionally with the ploidy.

Position	16	17	32	*56*	57
Haplotype 1	AACC	C	CCA	*T-*	-
Haplotype 2	AACC	A	CCA	TA	A
Haplotype 3	*A-*	-	*C*	TA	*C*

Position	16	17	32	56	57	Sum	Sum/ Total
Alleles/pos	2	3	2	2	3		
16AACC		0.67	0.5	1	0.67	2.83	0.63
16A-		0.33	0.5	0.5	0.33	1.67	0.37
					Total	4.5	
17A	0.5		0.5	0.5	0.33	1.83	0.33
17C	0.5		0.5	0.5	0.33	1.83	0.33
17-	0.5		0.5	0.5	0.33	1.83	0.33
					Total	5.5	
32CCA	0.5	0.67		1	0.67	2.83	0.63
32C-	0.5	0.33		0.5	0.33	1.67	0.37
					Total	4.5	
56TA	1	0.67	1		0.67	3.33	0.67
56T-	0.5	0.33	0.5		0.33	1.67	0.33
					Total	5	
57A	0.5	0.33	0.5	0.5	0	1.83	0.33
57C	0.5	0.33	0.5	0.5	0	1.83	0.33
57-	0.5	0.33	0.5	0.5	0	1.83	0.33
					Total	5.5	

Figure 30 Weighted connectivity table for toy example. In the top table we have the real 3 haplotypes with their observations. Assuming that we have reads 100 bp long, all positions are connected if they belong to the same color. Starting at 16, for each variation we compute their connection weighted by the number of alternatives per position with the other edges. At the end, we add the weights of all connections per examined edge and we normalize them by the total of all of their positions. The identified exclusive variations are highlighted.

Another idea would be to look at the degree of connectivity of the edges. On average, the exclusive variations are less connected than non-exclusives, since the exclusive observations can only have two edges with the immediate neighboring positions (one incoming and one outgoing), and only one connection with each other position (each position might have multiple variations but exclusive paths will only connect with one variation per loci). Unfortunately, in some cases, non-exclusive nodes can also be connected by two neighbor edges and traversed by multiple haplotypes that locally share the same alleles (**Figure 31**). This possibility decreases exponentially over lengths with higher numbers of observations. It is less likely to observe two haplotypes sharing the same variations over many contiguous positions than over just a few, but there will be segments where the reads will span only 2 variations, especially since we are dealing with short reads. In those cases, it is very likely that several haplotypes share two adjacent variations. But generally speaking, among all possible paths connecting an exclusive variation position and any other position, the connectivity of the exclusive observation will always have the minimal possible value.

In order to identify exclusive variations, we try to run the algorithm using an alternative measure of connectivity that weights the number of alternative alleles existing per position. For instance, in our toy example (**Figure 28 Top Table**) we can examine the connection of the two possible alleles in position 16: '16/AACC' and '16/A-'. Each of them could potentially be connected with three variations of position 17 ('17/C', '17/A' plus the deletion '17/-') and with two in position 32 ('32/CCA' and '32C'). We apply a different weight to the edges with position 17 nodes (which has 3 variations), than with those in position 32 (that only has 2). Node '16/AACC' connects with 2/3 nodes in position 17, but only with ½ of alleles in position 32. We do the same for all nodes connected by a read to any of the two alleles in position 16. Finally, for each node we add all of their weighted connections and we normalize by the sum of all weights per position, resulting in 0.63 for '16/AACC' and 0.37 for '16/A' which identifies the last one as being exclusive (**Figure 30**) since it has the minimum possible measure.

The intuition behind this measure is that connections with nodes in positions where two variations are possible should have a heavier weight than connections where three alternatives exist. A connection with an edge in a position where only two alternatives exist brings less information about the exclusiveness than edges connecting to a position where three alternatives exist. Those vertex with minimal scores are more likely to be exclusive.

Once the exclusive nodes are identified, we give each of them a color and extend following the principles 2 and 3 explained above. When an extension meets an edge which is already colored, the new edge and all its extensions take the color of the edge being extended, reducing at each step the number of colors present in the graph and the number of breaks in the final phasing estimation.

Phasing results

We tried our method using a 10 Kbp section of simulated chromosome I of *S.Pastorianus*. We tried to reconstruct the haplotypes using only the Illumina (2 x 100 bp) pair end reads with one library having 20bp overlaps (resulting in 180 bp pseudo-reads) and another one with 500 bp insert size, which limits the phasing output to contiguous variations that are separated by less than 698 bp. Some variation positions in our sample are separated by more than that distance so we were expecting some breaks in our haplotyping.

The analyzed section counts 161 variation positions. Since the contig is triploid, each position always contains one exclusive allele, so there are also 161 exclusive variations to be potentially identified; however, we are still limited by the size of the reads and the available libraries that we are using.

We only explored our method for phasing up to the exclusive edge coloring step. The method correctly identified 135/161 isolated variations and attributed a different color to each of them. We also implemented the color extending step along the rows and columns of the matrix, matching the colors of the interconnected isolated nodes to have the same value. The three longest unbroken haplotypes that were correctly reconstructed contained 80 variations, and they were found at the beginning of the sequence, after which the phasing broke. The first 29 exclusive variations (covering 974 bp) were correctly colored and extended and the first phasing break that creates a supplementary color occurs at position 980 bp. The rest of the genome, while having breaks in the color attribution, correctly clustered together 128/161 observations. Instead of the ideal three unbroken haplotypes, 17

breaks (different colors) were found. We stopped this line of investigation because our method to detect isolated nodes was incorrect.

Length of the simulated genome	10.000 bp
Number of Haplotypes	3
Total variation positions	161
Exclusive variations	161
Identified exclusive variations	135/161
Correctly identified exclusive variations	135/135
Correctly Colored exclusive variations (with breaks)	128
Correctly Colored exclusive variations (without breaks and in the top 3 bigger clusters)	80
Longest length correctly colored	974 bp
Final number of different colors (breaks)	17
Isolated variations (unique to have its color)	5
Erroneously clustered variations	7

Table 4 Phasing results on simulated genome of 10Kbp with three haplotypes. Exclusive observations were found with the weighted connectivity measure which doesn't work with higher ploidy and is not guaranteed to always find all exclusive nodes even in triploids.

The results of our method to identify exclusive variations with weighted connections work only on diploid and triploid contigs that have many positions connected within the span of a read, but is not guaranteed to always work otherwise. For instance, when more than one haplotype have the same local variations over a certain sequence, their edges will have the same weighted connectivity as those from a single haplotype traversing an exclusive set of variations.(**Figure 31**) The possibility of those segments existing is more likely to happen in regions where variations are very distanced from one another, therefore reducing the number of variation positions covered by the available reads.

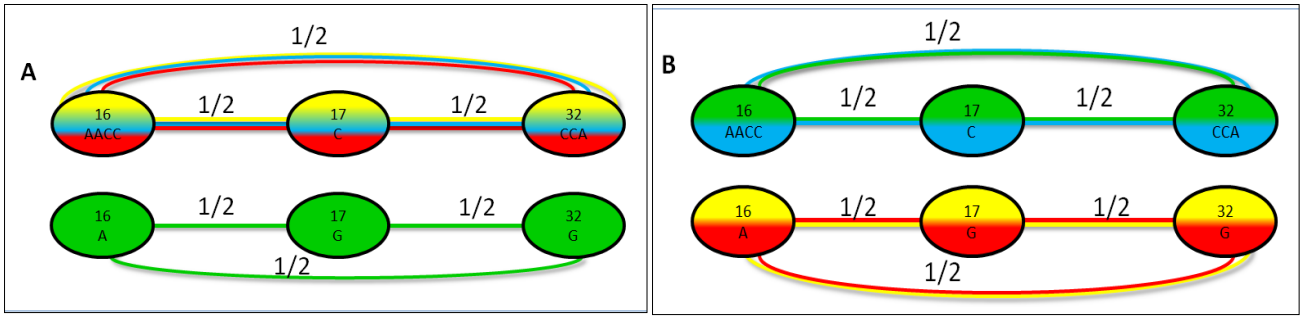


Figure 31 Two examples where the weighted connectivity of the nodes fails to detect the isolated variations. Both cases A and B reflect a segment with 4 haplotypes/colors traversing 3 positions 16, 17 and 32, each having 2 possible alleles. Case A has 3 colors traversing the same contiguous alleles in one path with the same connectivity weighted per position as the path with only one color (green). Both end up with the same minimum possible score of 0.5. Segment B also fails to identify the exclusive alleles. In this case, both paths have two colors traversing them, and both have the minimum possible score for those paths of 0.5.

Besides, the approach becomes increasingly unreliable with higher ploidy. Thus, we cannot consistently identify the isolated alleles with the weighted connectivity approach. At the very best we could develop a probabilistic approach that computes the likelihood of a given segment being exclusive.

It was misleading to use only a small sample for our tests with a single ploidy case of p=3. Because we didn't consider a wider range of cases, we failed to predict the limits of the actual implementation of the connectivity weighted by position method. But despite this error, the approach has the potential to be explored further. With a copy number of three, many isolated nodes were identified, and almost half of them were correctly colored together and without a break.

Phasing Discussion & Conclusion

Haplotyping with short reads remains by its intrinsic nature an optimization problem. Most likely, with the development of more accurate long read sequencing technologies, the problem will sooner or later be able to be solved with a *de novo* assembly. But even with the current available sequencing capacity of PacBio and Nanopore reads, this approach can probably be explored further. Finding a more reliable method to find the exclusive variations is a necessary first step in this direction. One of the conclusions learned from this exploration is the importance of doing extensive tests with higher ploidy from the beginning of the research. As we saw in both parts of this study, the complexity of polyploid genomes increases with the copy number.

But failing to find a proper methodology to identify exclusive nodes was not the only reason why the haplotyping wasn't completed. Nothing in our research concludes the possibility or impossibility of finding a reliable way to do this, and the errors reported for our method are a normal part of the creative and often frustrating process of any research. We had to abandon the lead and, at the same time, the possibility of finding a new approach to the aneuploid phasing, not so much because of this error but due to the time constraints of the research program and the practical problems involved in extending it.

There were, nevertheless, other challenges ahead. For our method to work, we need to have eliminated all errors in the reads. We managed to get rid of the sequencing errors by rejecting underrepresented variations with the help of the a priori information of the contig ploidy. But there are also alignment errors among the reads. Some of them are recurrent because sometimes a similar sequence can be aligned to a reference in multiple ways. For those that are constantly reported, the coverage filter becomes inefficient. Luckily, most of them are eliminated when compared against the variant calling, because they don't have any representation in the .vcf file and therefore get rejected. But some of them bypass both filters, especially when a false local alignment coincides with an existing alternative allele of another haplotype. These cases constitute a minority but when they happen, they cause color attribution conflicts that are hard to identify and that break the continuity in the extending color step.

At this current stage of the research, it is difficult to tell which of these problems are just temporary challenges that have solutions, and which ones are unsolvable obstacles that disqualify the whole approach. It seems that the adjacency matrix is a very useful way of storing and organizing the available read information and just a few steps away from the real solution. But solving haplotyping is a very complex problem, especially when the ploidy increases. Furthermore, with only short reads available, it can only be tackled as an optimization problem. For all those reasons, we decided instead to focus on improving PEDCA and finalizing it to deliver a ready-to-use tool, and simply reporting the efforts made in haplotyping.

A quick general conclusion with some personal reflections

Polyploid assembly remains a very wide challenge, even more so with short reads. We started this research by briefly comparing some available *de novo* assemblers. This step alone opens up its own field of study that is so broad that is often overtaken by the speed at which new sequencing technologies develop. It is most likely that long reads will provide many answers to some of the yet unsolved problems, especially the one of repetitive regions. Yet, at the present moment, the complexity of some aneuploid genomes still resists many of our approaches. We've seen how our data, mostly due to the coverage's standard deviation, becomes harder to analyze with depth of coverage analysis as the ploidy increases. Hybridization events, genomic content adaptations (like rapid gain and loss of heterozygosity), and small- and large-scale genome shifts and structural variations are also some of the assembly challenges that increase in complexity with an increase in copy number.

We have examined new alleys for haplotyping and described our results and failures. But the focus of this research was put on providing an improved tool for copy number estimation. The result is PEDCA, a fast depth of coverage analysis tool that not only examines the number of alignments to the reference genome but can also take into account the allele variations to provide supplemental analysis. It is easy to operate because it only requires either a .bam or a .sam file, but it outputs additional analysis if an optional .vcf file is provided. It works on any operational system that has a Java platform. It outputs a detailed estimation of each contig's ploidy and can detect fragments with copy number jumps within contigs.

The tool provides extra insight into intra contig ploidy composition thanks its allowing detailed estimation by fragments. This knowledge can be very helpful for better understanding not only phenotype expression but also the evolutionary history of hybridization events and the complex structural combinations that arise in that context. The tool also has the potential to study cancer cell biology since copy number mutations is a common trait that appears in tumor cells. In the cancer framework, identifying ploidy segments can be very helpful for detecting specific mutated regions of the genome that diverge from diploidy.

In order to achieve this work, many skills – both general and others that are much more specific to this particular research - had to be acquired. Computational biology has a very sharp learning curve and there are many challenges that students with no biological background have to face during their first years in the field.

For this particular research, it was important to deeply comprehend the details and the technicalities surrounding the processes of assembling, scaffolding, aligning, and variant calling. This also involved becoming familiarized very quickly with shell operated programs and the cohort of bioinformatics tools that goes with them. It was necessary to acquire vast quantities of high level multidisciplinary knowledge (genetics, mathematics, statistics and machine learning) and understand complex algorithms with enough detail to be able to implement and adapt them to the particular setting of the problem (EM algorithm, mixture of Gaussians fits, k-means…).

But this research has been much more than an academic journey. Behind the technical challenges and results hides a personal adventure with its own set of challenges and accomplishments of a very different nature. Coming from a very different and non-scientific background (holding a Master in

Social Communication and a Conservatory Degree in Theatre) and switching to the multidisciplinary field of Bioinformatics requires considerable effort, particularly for someone who is twenty years older than the average Master's student. Working internationally for over 15 years in cultural related fields, in Human Rights and International Cooperation can shape the habits, methodologies and ways of thinking and relating to the world in ways that are very different from the scientific paradigm. Having to also manage everyday practical needs (for example finances, social and personal life) requires supplemental efforts that cannot be properly evaluated solely from the academic perspective.

During this study, it was very satisfactory to see the impressive work of the people at the Bioinformatics Laboratory of Delft applying very complex techniques to solve problems that can have such a tremendous impact on society. Computational biology has been driving the knowledge revolution for the last 20 years, bringing fast dramatic changes and profound implications as we increase our understanding of genetics' complexity. The simplest developed tools have the potential to help biologists make advances that before would have taken them years to solve with pure wet lab labor. Inevitably, these applications will often be guided and influenced by the needs of other areas, very often with intentions that have no scientific interest and no benefits for society as a whole, but provide benefits only to private interests.

It is the author's opinion that great care must be taken when choosing the direction in which of all this power is applied. The *Trypanosoma cruzi* genome provided a meaningful context for the application of PEDCA. While not having a strong personal preference for or against modifying a yeast genome so that a industrial beer producer can reduce its costs, this particular author feels much more personal satisfaction knowing that the fruit of such efforts can instead bring insights that can potentially lead to prevention, treatments and cures for diseases that affect thousands of people around the world.

It is in this direction that hopefully he will advance and deepen in his future steps in bioinformatics.

Personal Acknowledgements

The author takes this opportunity to thank the entire academic panel from Delft and Leiden, which has accompanied him during these two and a half years in his initiation into the dark arts of computational biology, pushing him beyond limits he previously didn't even knew he had. Special thanks to Thomas Abeel and his pedagogic professionalism and human management skills, who guided this research to meet rigorous standards that the author could never have attained alone. Thank you to Marcel Reinders and all of his staff for managing the Bioinformatics Delft lab with the art of constantly facilitating the exchange of knowledge within a team with very diverse backgrounds. Thank you to all of the members of Delft's Biolab for showing such an example of excellence in their skills and work. Particular thanks to Marcel Van den Broek for his always warm availability to share his work with generosity and passion. Thank you also to Erwin Bakker for contributing to the supervision of this work from Leiden.

The author also wishes to personally thank his parents for all their loving support, without which this journey could have never come to be. A very special thanks to Catherine Tylke for her support and assistance in reading and correcting the final version of this document. Finally, thank you to the organization 'House of Indians' for the inspiration and help received over the course of this study.

References

Aguiar, D., & Istrail, S. (2012). HapCompass: A Fast Cycle Basis Algorithm for Accurate Haplotype Assembly of Sequence Data. *Journal of Computational Biology*, *19*(6), 577–590. http://doi.org/10.1089/cmb.2012.0084

Alkan, C., Sajjadian, S., & Eichler, E. E. (2011). Limitations of next-generation genome sequence assembly. *Nature Methods*, *8*(1), 61–65. http://doi.org/10.1038/nmeth.1527

Berger, E., Yorukoglu, D., Peng, J., & Berger, B. (2014). HapTree: A novel bayesian framework for single individual polyplotyping using NGS data. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8394 LNBI*(3), 18–19. http://doi.org/10.1007/978-3-319-05269-4_2

Bradnam, K. R., Fass, J. N., Alexandrov, A., Baranay, P., Bechner, M., Birol, I., … Korf, I. F. (2013). Assemblathon 2: evaluating *de novo* methods of genome assembly in three vertebrate species. *GigaScience*, *2*(1), 10. http://doi.org/10.1186/2047-217X-2-10

Butler, J., MacCallum, I., Kleber, M., Shlyakhter, I. a., Belmonte, M. K., Lander, E. S., … Jaffe, D. B. (2008). ALLPATHS: *De novo* assembly of whole-genome shotgun microreads. *Genome Research*, *18*(5), 810–820. http://doi.org/10.1101/gr.7337908

CDC. (2011). Center for Disease Control: PIV. http://doi.org/doi:10.1037/e548442006-001

Das, S., Vikalo, H., Clark, A., Gibbs, R., Belmont, J., Hardenbol, P., … Vandenberghe, L. (2015). SDhaP: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genomics*, *16*(1), 260. http://doi.org/10.1186/s12864-015-1408-5

David Gilbert. (n.d.). JFreeChart. Retrieved March 31, 2017, from http://www.jfree.org/jfreechart/

Earl, D., Bradnam, K., St. John, J., Darling, A., Lin, D., Fass, J., … Paten, B. (2011). Assemblathon 1: A competitive assessment of *de novo* short read assembly methods. *Genome Research*, *21*(12), 2224–2241. http://doi.org/10.1101/gr.126599.111

El-Sayed N. et al. (2005). The Genome Sequence of *Trypanosoma cruzi*, Etiologic Agent of Chagas Disease. *Science 15 Jul 2005*, *309*(5733), 409–415. http://doi.org/10.1126/science.1112631

Escalona, M., Rocha, S., & Posada, D. (2016). A comparison of tools for the simulation of genomic next- generation sequencing data Europe PMC Funders Group. *Nat Rev Genet*, *17*(8), 459–469. http://doi.org/10.1038/nrg.2016.57

Galardini, M., Biondi, E. G., Bazzicalupo, M., & Mengoni, A. (2011). CONTIGuator: a bacterial genomes finishing tool for structural insights on draft genomes. *Source Code for Biology and Medicine*, *6*(1), 11. http://doi.org/10.1186/1751-0473-6-11

Garcia, V., & Nielsen, F. (n.d.). jMEF. Retrieved March 31, 2017, from http://vincentfpgarcia.github.io/jMEF/index.html

Genomics, B., Weatherly, D. B., Boehlke, C., & Tarleton, R. L. (n.d.). Chromosome level assembly of the hybrid *Trypanosoma cruzi* genome. http://doi.org/10.1186/1471-2164-10-255

Gurevich, A., Saveliev, V., Vyahhi, N., & Tesler, G. (2013). QUAST: Quality assessment tool for genome assemblies. *Bioinformatics*, *29*(8), 1072–1075. http://doi.org/10.1093/bioinformatics/btt086

Huang, W., Li, L., Myers, J. R., & Marth, G. T. (2012). ART: a next-generation sequencing read simulator. *BIOINFORMATICS APPLICATIONS NOTE*, *28*(4), 593–59410. http://doi.org/10.1093/bioinformatics/btr708

Ley, V., Andrews, N. W., Robbins, E. S., & Nussenzweig, V. (n.d.). AMASTIGOTES OF *TRYPANOSOMA CRUZI* SUSTAIN AN INFECTIVE CYCLE IN MAMMALIAN CELLS.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., … Subgroup, D. P. (2009). The Sequence Alignment/Map format and SAMtools. *BIOINFORMATICS APPLICATIONS NOTE*,

25(1610), 2078–2079. http://doi.org/10.1093/bioinformatics/btp352

Li, R., Zhu, H., Ruan, J., Qian, W., Fang, X., Shi, Z., … Wang, J. (n.d.). *De novo* assembly of human genomes with massively parallel short read sequencing. http://doi.org/10.1101/gr.097261.109

Libkind, D., Hittinger, C. T., Valerio, E., Goncalves, C., Dover, J., Johnston, M., … Sampaio, J. P. (2011). Microbe domestication and the identification of the wild genetic stock of lager-brewing yeast. *Proceedings of the National Academy of Sciences*, *108*(35), 14539–14544. http://doi.org/10.1073/pnas.1105430108

Margarido, G. R. A., & Heckerman, D. (2015). ConPADE: Genome assembly ploidy estimation from next-generation sequencing data. *PLoS Comp Biol*, *11*(4), e1004229. http://doi.org/10.1371/journal.pcbi.1004229

Motazedi, E., Finkers, R., Maliepaard, C., & de Ridder, D. (2016). Exploiting Next Generation Sequencing to solve the Haplotyping puzzle in Polyploids: a Simulation study. *bioRxiv*. http://doi.org/10.1101/088112

Myers, E. W. (1995). Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology : A Journal of Computational Molecular Cell Biology*, *2*, 275–290. http://doi.org/10.1089/cmb.1995.2.275

Myers, E. W. (2005). The fragment assembly string graph. *Bioinformatics*, *21*(Suppl 2), ii79-ii85. http://doi.org/10.1093/bioinformatics/bti1114

Nijkamp, J. F., van den Broek, M. a, Geertman, J.-M. a, Reinders, M. J. T., Daran, J.-M. G., & de Ridder, D. (2012). *De novo* detection of copy number variation by co-assembly. *Bioinformatics (Oxford, England)*, *28*(24), 3195–202. http://doi.org/10.1093/bioinformatics/bts601

Nter Klambauer, G., Schwarzbauer, K., Mayr, A., Clevert, D.-A., Mitterecker, A., Bodenhofer, U., & Hochreiter, S. (n.d.). cn.MOPS: mixture of Poissons for discovering copy number variations in next-generation sequencing data with a low false discovery rate. http://doi.org/10.1093/nar/gks003

Pevzner, P. a, Tang, H., & Waterman, M. S. (2001). An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences of the United States of America*, *98*(17), 9748–9753. http://doi.org/10.1073/pnas.171285098

The Polyploidy Portal. (n.d.). Retrieved from http://polyploidy.org/index.php/Parade_of_Polyploids

van den Broek, M., Bolat, I., Nijkamp, J. F., Ramos, E., Luttik, M. A. H., Koopman, F., … Daran, J.-M. (2015). Chromosomal Copy Number Variation in Saccharomyces *pastorianus* Is Evidence for Extensive Genome Dynamics in Industrial Lager Brewing Strains. *Applied and Environmental Microbiology*, *81*(18), 6253–6267. http://doi.org/10.1128/AEM.01263-15

Wagner, A. (n.d.). Rapid Detection of Positive Selection in Genes and Genomes Through Variation Clusters. http://doi.org/10.1534/genetics.107.074732

Walker, B. J., Abeel, T., Shea, T., Priest, M., Abouelliel, A., Sakthikumar, S., … Earl, A. M. (2014). Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PloS One*, *9*(11), e112963. http://doi.org/10.1371/journal.pone.0112963

Wetzel, J., Kingsford, C., & Pop, M. (2011). Assessing the benefits of using mate-pairs to resolve repeats in *de novo* short-read prokaryotic assemblies. *BMC Bioinformatics*, *12*(1), 95. http://doi.org/10.1186/1471-2105-12-95

Xie, C., & Tammi, M. T. (2009). CNV-seq, a new method to detect copy number variation using high-throughput sequencing. http://doi.org/10.1186/1471-2105-10-80

PEDCA Tutorial (Ploidy Estimation by Dynamic Coverage Analysis)

PEDCA is a ploidy estimation algorithm that infers copy number of the contigs submitted as input based on the read coverage that aligns to them. It only requires as an input an alignment file in .bam or .sam format of a library or set of libraries aligned to a reference file of the contigs that will be estimated.

Pre-processing the data (5 steps)
We need to align the reads against a reference.

Step 1. Index your reference.
Example using bwa (all command in one single line):
<*path_to_bwa_aligner*>/bwa index -a bwtsw <*path_to_reference_file/your_reference.fasta*>

Step 2. Align your reads to your reference
Example using bwa and paired end reads (all command in one single line):

<*bwa_aligner_path*>**/bwa mem** <*path_reference_file/your_reference.fasta*> <*readsPath/readsPairEnd1.fasta*> < *readsPath /readsPairEnd2.fasta*> > <*destination_folder* /*example.sam*>

Step 3. You might want to transform your .sam file into a .bam format
Example using samTools (all command in one single line):

<*samToolsPath*> **/samtools view -Sb** <*destination_folder* /*example.sam*> > <*destination_folder* /*example.bam*>

PEDCA just accepts one input file. If you have several libraries you can put all your .bam files in a folder (or create a folder with symbolic links to all files you want to merge) and then:

<*samToolsPath*>**/samtools merge** <*bam_destination_folder*/finalBamFile.bam> ***.bam**

Step 4. Sort the .bam/.sam file
Example sorting a .bam file using samTools (all command in one single line):

<*samToolsPath*> **/samtools sort -o** <*destination_folder* /sorted_*example.bam*> **-O** .bam **-T** <*temp_folderPath*/*tempName*> <*destination_folder* /*example.bam*>

Step 5. Index the sorted .bam/.sam file
Example indexing a sorted .bam file using samTools (all command in one single line):

<*samToolsPath*> **/samtools index** <*destination_folder* /sorted_*example.bam*>

Using PEDCA

PEDCA has been designed to require minimal parameterization. It works by running a sliding window over the genome and measuring the average depth of coverage inside each bin. Most of its parameters are dependent on the window length and have default values that allow PEDCA to function on contigs > 500 bp and < 2.000 Kbp. Nevertheless, because each genome has its own particular characteristics, it is possible to tune in the rest of the parameters. Here is a list of those options and how they influence the output.

At any moment you can obtain the following guide using: **`java –jar PEDCA.jar -help`**
(You can also use '**`–h`**' or '**`help`**')

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
PEDCA -help:

USAGE:    java –jar PEDCA.jar  -p <project name> -i < input sam/bam File> -o <output
Folder> <<OPTIONAL PARAMETERS>>

REQUIRED PARAMETERS:
-p (project name)          – (String)  Prefix used to generate the results file.
-i (input file)            - (Pathway to .bam/.sam file)   Pathway to the input file
containing the alignment file. Must be a .bam or .sam file
-o (output Folder)          - (String)   Pathway to the auto-generated output folder
that will contain the results

OPTIONAL PARAMETERS:
-m (multi Run)             - (no parameters) Runs a preselected set of default window
lengths {500,750,1000,2000,3000}
-w (windows length)        - (int)      Length of the sliding window, to measure the
coverage inside  contig. Default 500 bp
-c (coverage rate)          - (int)     Rate factor for the coverage sampling in the
Read count distribution. Default 100. The smaller it is, the less bins are sampled
-k (mode smoother window) - (int)       Number of points over which the ploidy
estimation is smoothed. The mode over k numbers of windows is used to average the
values of the bin. Default=49
-s (significant min)        - (double) Threshold to consider a cluster peak in the
read count to be significant. Default 0.1
-b (fitter bin factor)       - (double)  Affects the number of bins used to FIT the
read count distribution. Default 2.5; Recommended between min=2.0 and max=4.0
-v (allele frequencies)    - (Pathway to .vcf file)  Pathway to the file containing
the variant calling. Must be a .vcf file
-d (coverage data to use)     - (double) Fraction of coverage data that is used in
the read count distribution to infer the different ploidies and their ratio. Values
between 0 and 1. Default 0.97
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Downloading PEDCA https://github.com/AbeelLab/Pedca

REQUIRED PARAMETERS:

The first three arguments are required for PEDCA to function:

```
-p <project name> -i < input sam/bam File> -o <output Folder>
```

PEDCA creates a folder named by the concatenation of the project name and the size of the window length at the output pathway indicated by the user. The output has the following structure:

./<OutputFolderPath>
- ./BaseCall
  - . BaseCallHistogramCluster_1.jpg
  - . BaseCallHistogramCluster_2.jpg
  - . Matrix1stCluster.vcf
  - . Matrix2ndCluster.vcf
- ./<Project Name<*wl*>>
  - *./Ploidy_Estimation_Charts*
  - *.PEDCA*<Project Name<*wl*>>*PloidyEstimation.txt*
  - *.PEDCA*<Project Name<*wl*>>*PloidyEstimation_2nd_Round_.txt*
  - *. readsDistribution.jpg*
  - *.readsDistributionFittedFINALRESULT.jpg*

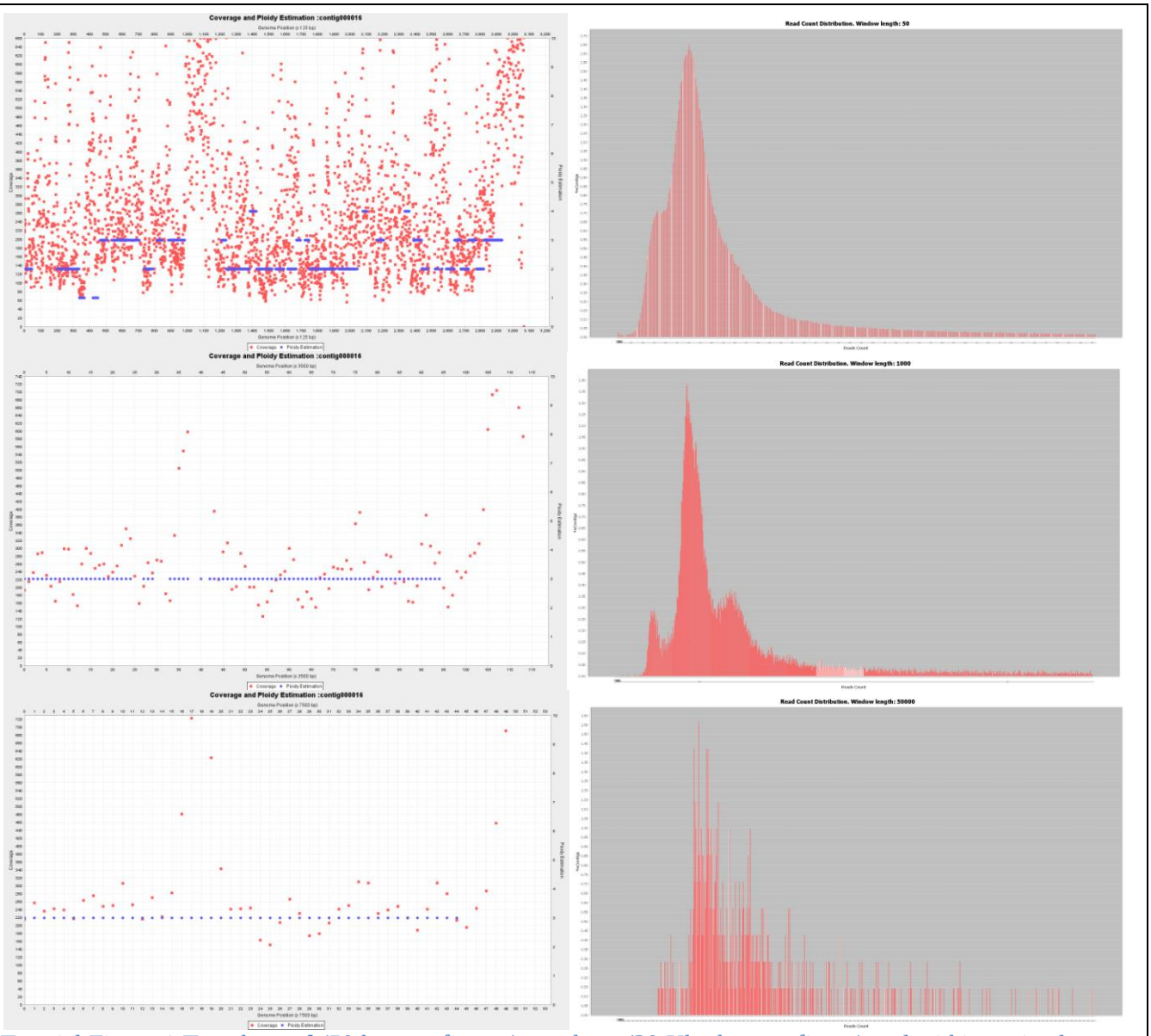
OPTIONAL PARAMETERS:

```
-w <window length>
```

It is worth noting that the window length (*wl*), despite being the main parameter in PEDCA, is not a mandatory field. If no other preference is indicated, PEDCA runs with the default value of *wl*=500 bp. Even if the parameter is not required, the advantage of using PEDCA is to have a customizable window length; therefore it is strongly recommended to use it with different values and compare the results.

A short *wl* provides more coverage data points to estimate the ploidy; you may want to shorten the *wl* if your ploidy estimation plot is too discontinuous or if it doesn't have much coverage information to support a reliable estimation (**Tutorial Figure 1**). On the other hand, you may want a larger *wl* if your coverage/estimation plot looks overcrowded with too much variation in the coverage data, which leads to a fragmented discontinuous copy number estimation (**Tutorial Figure 1**). The minimum size of the window is 16 bp.

The *wl* also affects the sampling in the read count distribution. If it is too big, it will lead to an irregular sampling with unrecognizable clusters and false cluster ratios (**Tutorial Figure 1**). If it is too small, the read count distribution will have its clusters merged together with long and thick tails that might hide undetected peaks (**Tutorial Figure 1**).



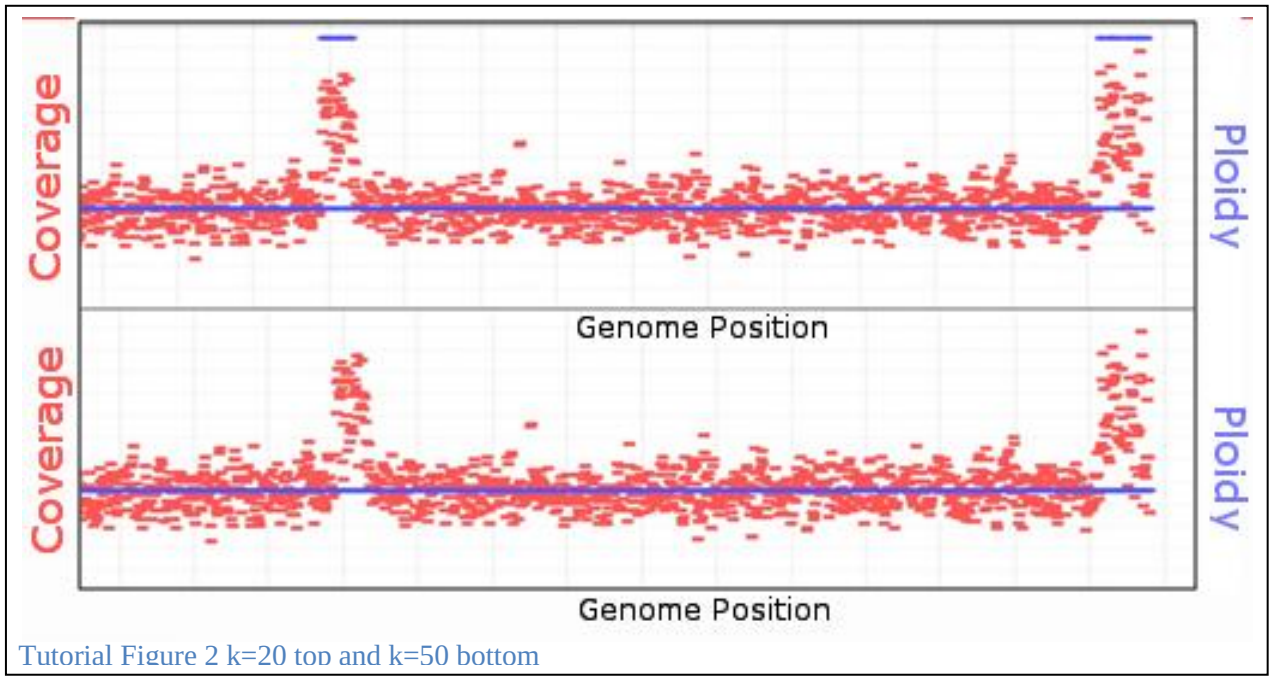
Tutorial Figure 1 Too short ***wl*** (50 bp; top figures), too long (30 Kbp bottom figure) and within optimal range (3 Kbp center figure)

`-v <Pathway to .vcf file>`

If this option is selected and a .vcf file submitted, a folder named BaseCall is also created at the same address, containing the allele frequencies plots and a matrix with the positions and frequencies of each base (order A,C,G,T).

`-k <mode smoother window>`

The coverage data has a certain degree of variation that we don't want to see reflected in the copy number estimation. In order to avoid undesired jumps in the ploidy plot, the points are averaged by the mode value over a bin of length *k*. If *k* is too small, it might lead to fragmented ploidy estimation in regions with noisy coverage (**Tutorial Figure 2 top**). The continuity is smoothed with the default *k* value of 50 bp (**Tutorial Figure 2 bottom**). The correct length of *k* depends on the required precision and can be parameterized. If *k* is too big, it might lead to the non detection of regions with different ploidies (i.e. large structural variations found in hybrid genomes)



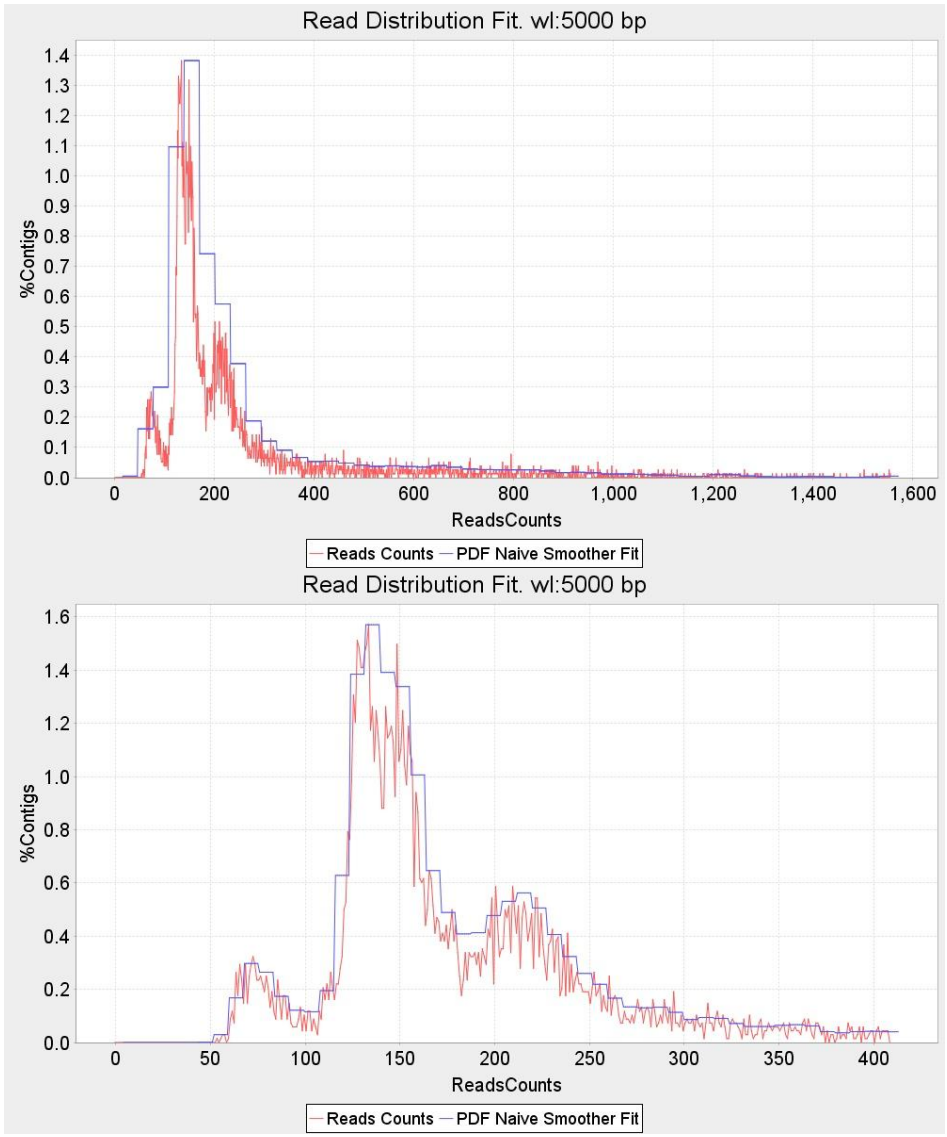
Tutorial Figure 2 k=20 top and k=50 bottom

`-m  <multiple window lengths run>`

This parameter enables multirun mode. Instead of running PEDCA with one single *wl* value, it automatically runs it five times with the preset values {500, 750, 1000, 2000, 3000} and outputs the respective results to the output folder. These values work well for contigs larger than 500 bp and up to 2.000 Kbp.

`-d <coverage portion of data to use (lower values)>`

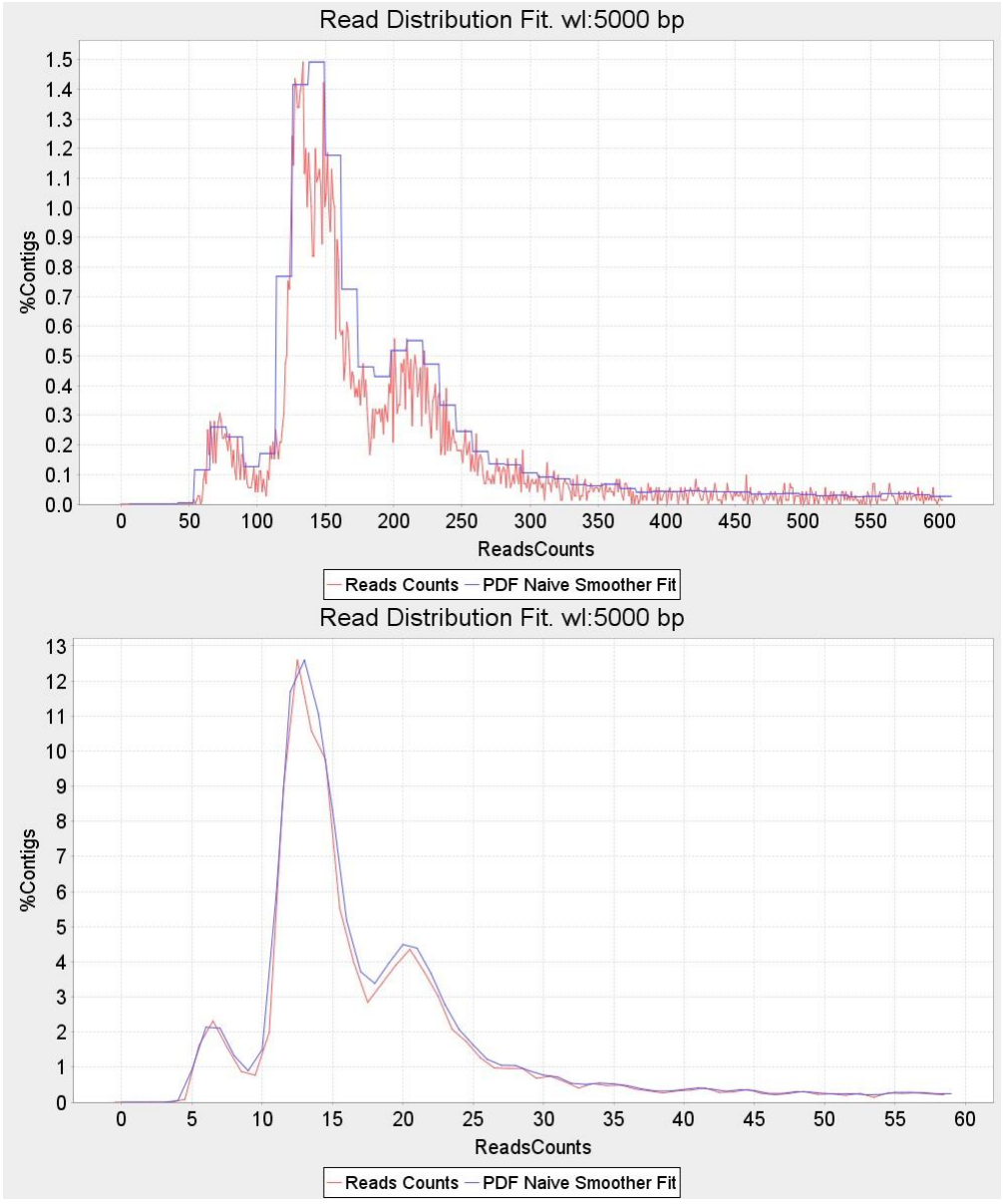
In order for PEDCA to correctly fit and identify the read count clusters, it is important to select the correct domain of the plot. By default, PEDCA keeps 0.97 of the data and rejects the top values, which doesn't affect the fitting of the read count PDF. Nevertheless, some highly noisy data sets might have long tails on their last cluster with no significant ploidy information. This long tail might compress the significant clusters to a very small region of the domain, making it difficult to differentiate them and find their correct ratio. In those cases it can be very helpful to lower the fraction of data to use (and therefore narrow the domain). Values can range from 0 to 1 (**Tutorial Figure 3**).



Tutorial Figure 3 The top plot uses a fraction 0.99 of the total data which makes its clusters hard to be identified. The bottom plot reframes the domain by using 0.85 creating the necessary space among the clusters to identify them. Notice that this modification might affect also the range values potentially needing a tuning in the *significant_min* parameter (-s)

`-c <coverage rate>`

The coverage rate is the definition with which the read count distribution is drawn. It affects the number of bins in the plot. The default value is 100. In some cases, when the plot is too irregular and sawed, it can be convenient to reduce this rate to have a fit that doesn't identify false peaks **(**Tutorial Figure 4**)** . The bins might merge clusters if the sampling rate is too low, so it's recommended to use this option with caution.



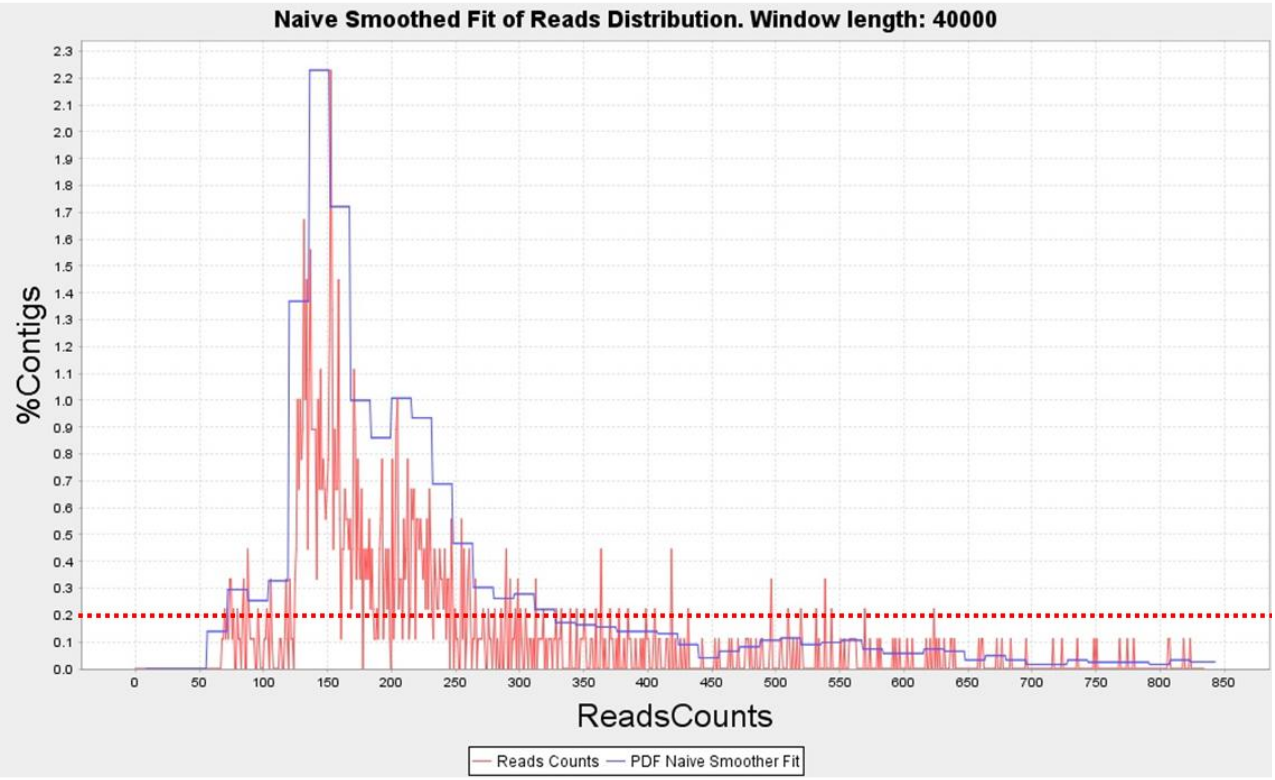
Tutorial Figure 4 Two read count distribution of the same data set where only the coverage rate changes. Left c=250, right c=15

`-s <significant min>`

When the read count is fitted, only peaks detected above a certain threshold are taken into account; otherwise insignificant oscillation peaks could be considered as clusters. The default value is preset to s= 0.1 % of the normalized number of windows for a given read count. For some genomes, this value might be too big and real clusters could be missed with a potential misinterpretation of the correct cluster ratio. On the other hand, if the distribution has a long tail with isolated values that are not considered clusters, it is important to raise the threshold to ignore false peaks in that region that would also jeopardize finding the appropriate cluster ratio.

In the example in **Tutorial Figure 5**, many micro peaks are detected in the long tail of the distribution. With a 0.2 significant minimum, all peaks below the red line are discarded. If the default value was used instead, an error message would be displayed because the peaks' ratio would not make sense:

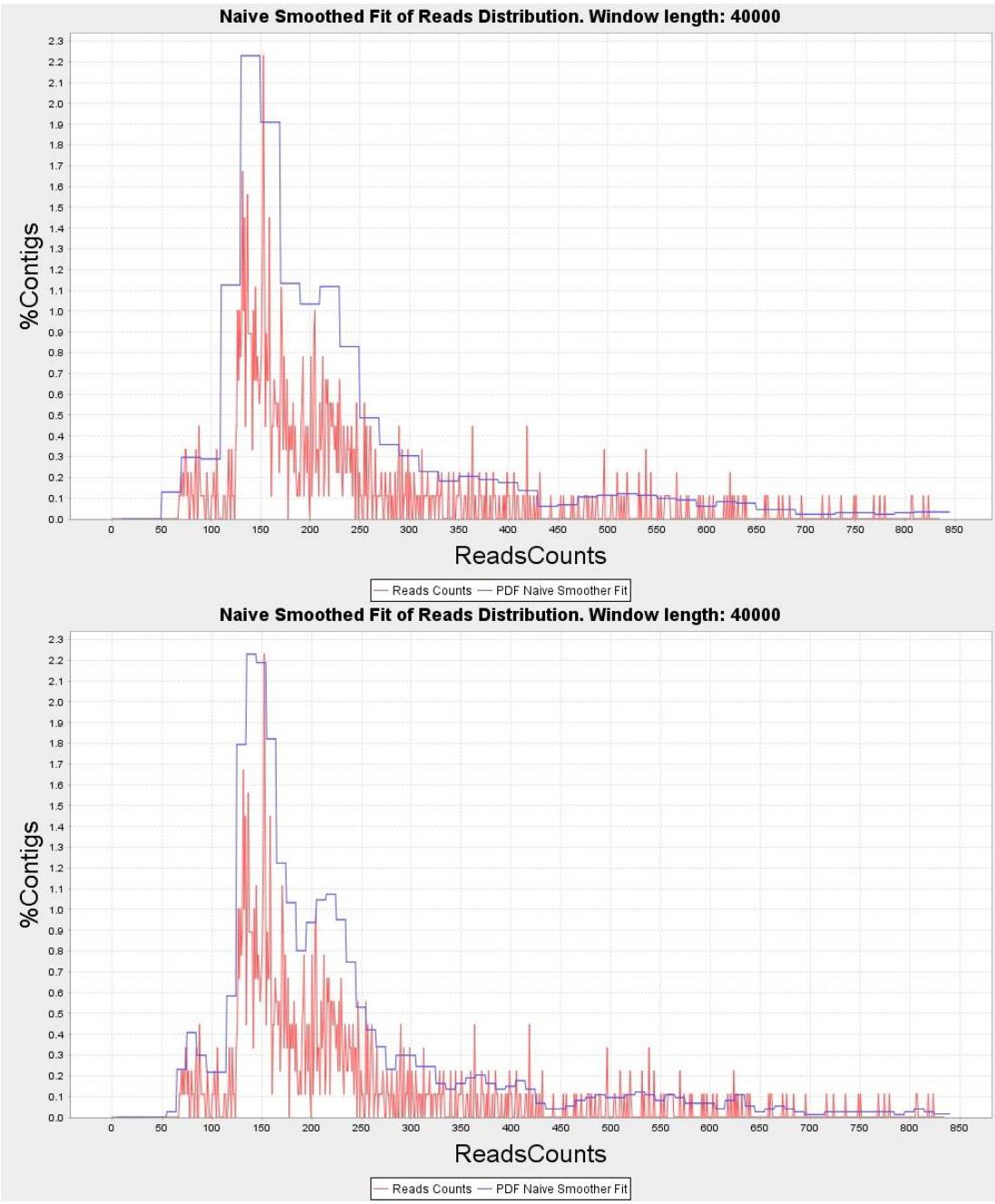
++++++++++ bestScore.candidateUnit: No CN mixture was able to satisfy the constraints. Result == null



Tutorial Figure 5 The red dots line indicates the significant min value (here 0.2%) below which peaks will not be taken into consideration

`-b <number of fit bins>`

With the default value (2.5) PEDCA fits the read count distribution with 25 bins, that is 2.5 x the maximum number of ploidies that PEDCA can detect. That number is adapted to detect a few clusters that are not very spread out over the x-axis of the read count distribution. If the clusters are far away from each other, a higher number might better fit the distribution. It is recommended to remain between the values min=2.0 and max=4.0



Tutorial Figure 6 Two read count distribution fits where only the fit bin factor rate changes. Top b=2.0, bottom b=4.0

Supplement material

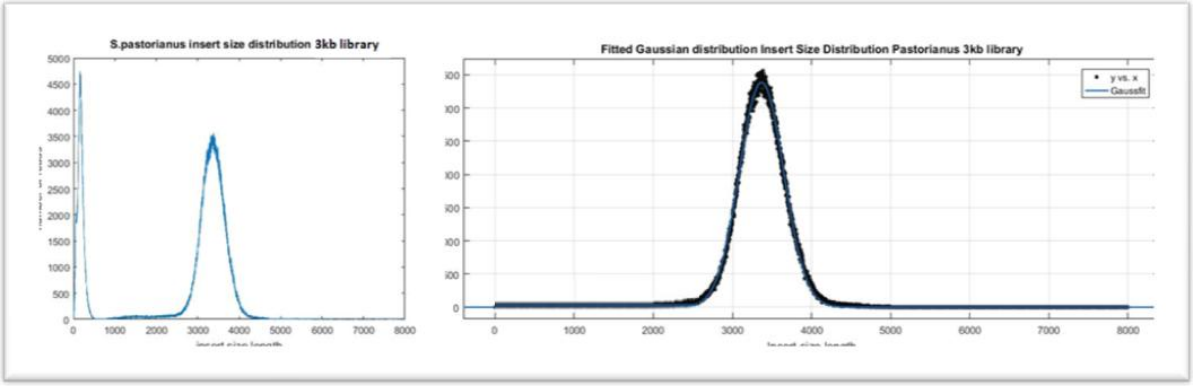
S.Pastorianus CBS1483 Illumina & Simulated Data stats

Insert size	Number of reads	Reads length
180bp	$9.26 * 10^6$	100 bp
500bp	$17.2 * 10^6$	100 bp
3kb	$40.1 * 10^6$	50 bp
8kb	$36.3 * 10^6$	50 bp

Supplemental Table 1 S.pastorianus CBS1483 Sequenced data statistics.

Libraries used to obtain the CBS1483 *S.Pastorianus* reference used in this study are the same than those used by (van den Broek et al., 2015). The same libraries were used to obtain the variations that were introduced into our five simulated chromosomes and their respective haplotypes. We also used the libraries statistics to build our simulated reads with the same values (length, number of reads, insert size and insert size distribution)

Estimating insert size and insert size standard deviation from the *S.Pastorianus* CBS1483 libraries to recreate simulated reads of same characteristics



Supplemental Figure 1 Insert size mean and standard deviation of 3kb library. After fitting a Gaussian distribution to the insert sizes of the library with Matlab, we obtain the mean and its standard deviation. The first cluster at the left side of the insert size distribution is ignored, considered to be error reads.

Method used to estimate insert size of our simulated reads.

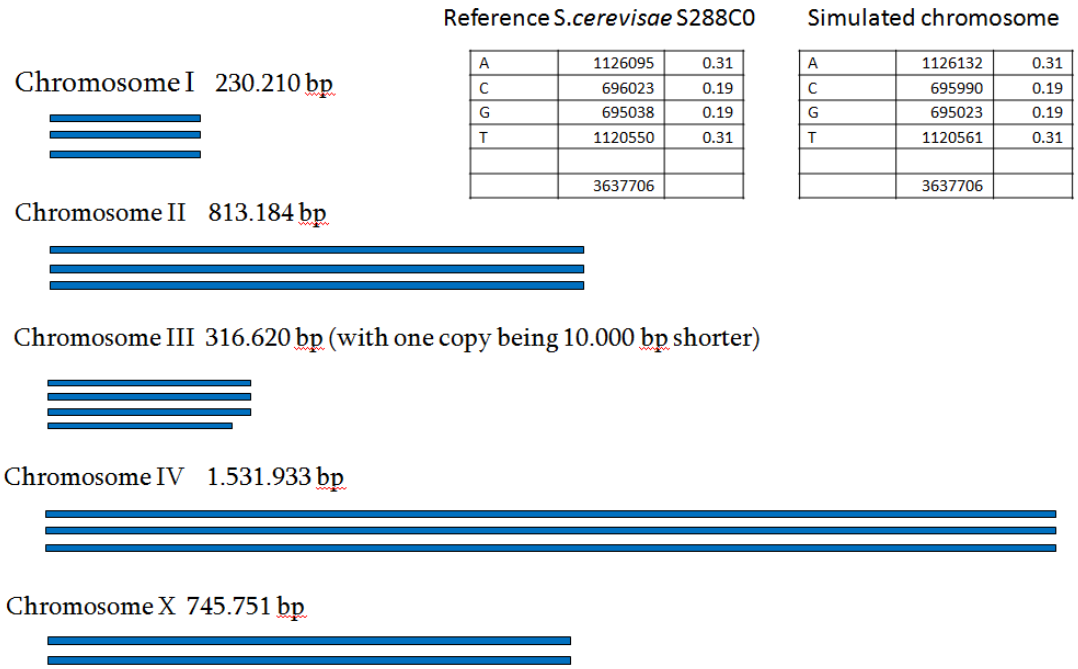
1) Both ends of BaseclearCBS1483 paired-end data is aligned against the CBS1483 reference genome (providing both fastq files to the bwa-0.7.13 aligner).

2) The information from the SAM/BAM files (TLEN field=fragment size) is extracted selecting only the number in the .bam's 9th column of those paired-end read pairs that are uniquely aligned against the genome.
3) The mean of the distribution of TLEN numbers generated in step 2 is computed. ie: mean fragment size(data)=3364
4) The length of the sequenced end pair read (2x50 bp or 2x100 bp) is substracted from the mean fragment size to get the insert size. ie: 3364-(2x50)=3264 μ(data)=3264 std(data)= 9.78% of μ

# Positions	3637706	100%
# Passes	3313517	91.03%
# No Passes (Variations)	326272	8.96%
# Amb (SNP's)	16429	0.45%
# Del	92814	2.55%
# Low Coverage	206209	5.67%
# Del/LowCov	8336	0.23%
# Del/Amb	2371	0.07%
# Amb/LowCov	104	0.3%
# Del/Amb/LowCov	9	...

Supplemental Table 2 Variant calling statistics on pastorianus reads over cerevisiae reference (chromosomes1-4 &10)

Composition of Simulated chromosomes



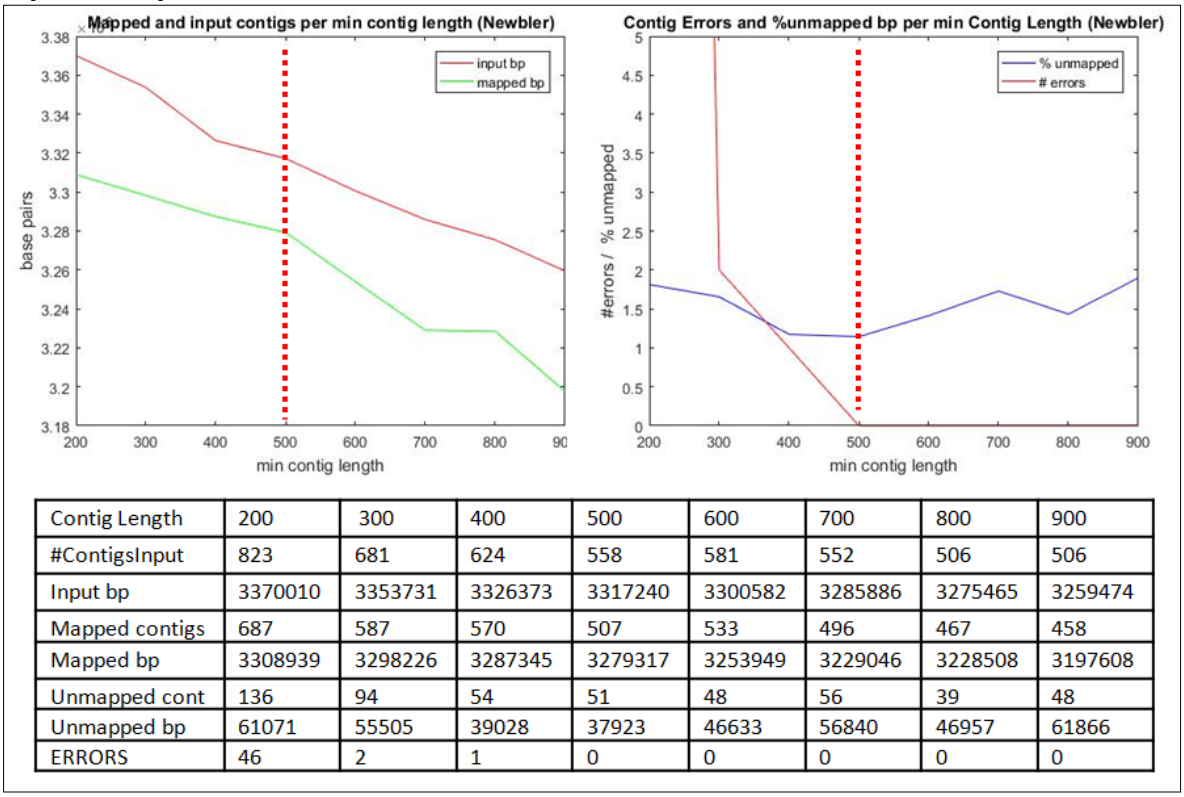
Supplemental Figure 2 Composition of the Five simulated chromosomes and composition comparison of base content between the real S.cerevisiae chromosomes and the simulated ones

Optimizing the Newbler assembly

We run Newbler on our five simulated CBS1483 chromosomes with the default parameters. Without scaffolding, our assembly resulted in 825 contigs, many of them being shorter than 1.000bp. We want to retain the optimal amount of that assembly rejecting the shortest contigs. We measured the optimal minimal contig length to consider in order to minimize the number of assembly errors (contiguous regions that cannot be mapped together to the reference) and maximize the amount of contigs that can be aligned to the reference. Contigs shorter than the minimal contig length are then rejected.

Supplemental Figure 3 top right shows that keeping only contigs > 500bp results in zero assembly errors. It also shows (Supplemental Figure 3 top left) that it minimizes the amount of total assembly that cannot be mapped to the reference (difference between the length of the input assembly and the amount that actually maps to the reference) in. This assembly retains 507/558 contigs. The optimal minimum contig length, corresponds with the shortest contig of CBS1483 assembly by (van den Broek et al., 2015), which was also assembled with Newbler.

We reduced the number of contigs to 286 scaffolds with SSPACE, but we could not scaffold without any assembly errors.



Contig Length	200	300	400	500	600	700	800	900
#ContigsInput	823	681	624	558	581	552	506	506
Input bp	3370010	3353731	3326373	3317240	3300582	3285886	3275465	3259474
Mapped contigs	687	587	570	507	533	496	467	458
Mapped bp	3308939	3298226	3287345	3279317	3253949	3229046	3228508	3197608
Unmapped cont	136	94	54	51	48	56	39	48
Unmapped bp	61071	55505	39028	37923	46633	56840	46957	61866
ERRORS	46	2	1	0	0	0	0	0

Supplemental Figure 3 Top left figure shows the difference between the total amount of Newbler's assembly's sequence (input bp –magenta-) and the amount of those contigs that maps to the control reference of our simulated data (mapped bp –green-). The minimum distance between them corresponds to the assembly that most reduces its unmapped regions (at min contig length = 500bp). Top right figure shows that the same value reduces the number of assembly errors to zero (magenta) while minimizes the % the assembly that remains unmapped (blue). Results were obtained using Quast (Gurevich et al., 2013)

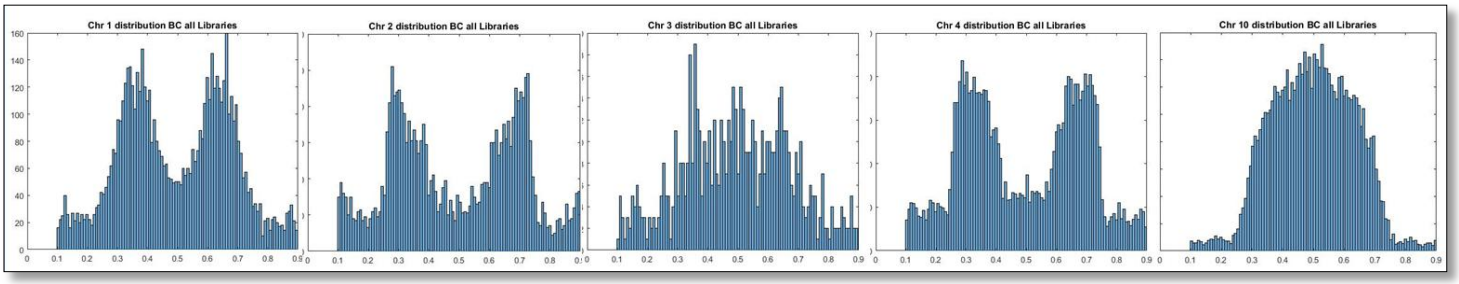
Magnolya result on our 5 simulated chromosomes (Extract from Magnolya output).

Contig name		Alignment Reference	start	end	length	Copy Estimation
contig00111	0.0	Chromosome_2_Reference	628606	637965	9360	1
contig00110	0.0	Chromosome_2_Reference	206032	215436	9405	1
contig00113	0.0	Chromosome_4_Reference	83547	92745	9199	1
contig00112	0.0	Chromosome_3_Reference	254722	263945	9224	1
contig00115	0.0	Chromosome_2_Reference	169139	178232	9094	1
contig00114	0.0	Chromosome_10_Reference	188193	197312	9120	1
...						
contig00203	0.0	Chromosome_10_Reference	391117	396730	5614	1
contig00202	0.0	Chromosome_1_Reference	70055	75680	5626	1
contig00530	0.0	Chromosome_10_Reference	737643	738223	581	1
contig00471	0.0	Chromosome_1_Reference	155918	156834	917	1

Supplemental Table 3 Our simulated dataset contained contigs with copy number raging from 2 to 4. Magnolya estimated a copy number of 1 for all contigs.

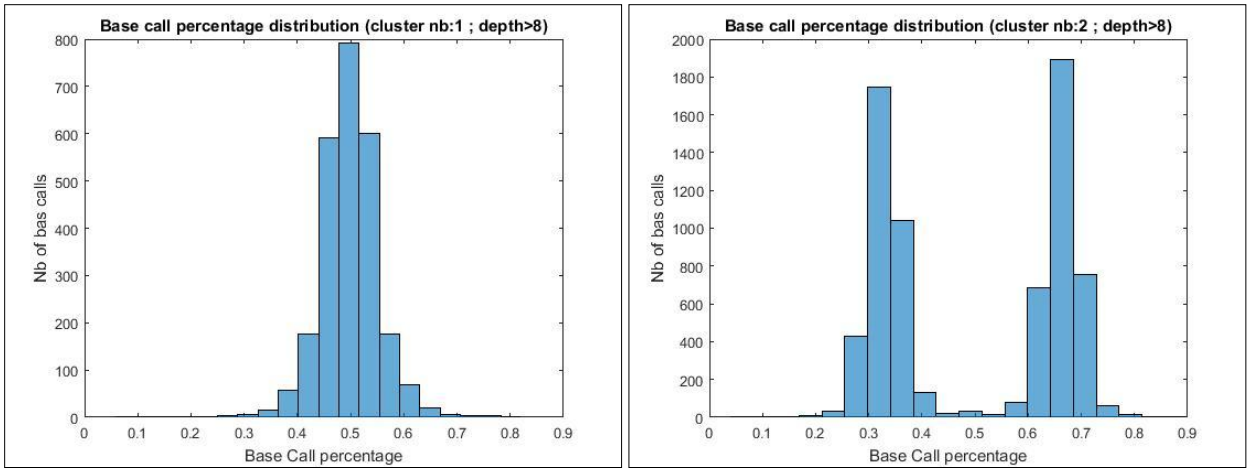
Using the allele frequencies to disambiguate the ploidy ratio.

Aligning the available libraries against a given reference allows measuring the variations that were lost in the consensus assembly. The shape of the allele frequencies distribution can help identify the copy number of each contig. A plot with bell shape centered around the 0.5 value, reflects that all bases are around the 50% chance of being in any of two alternative alleles, indicating a copy number of two for that chromosome (like chromosome 10 in Supplemental Figure 4) A plot with two peaks around 0.33 and 0.66, indicates base call present in either the 1/3 or 2/3 ratios, indicating a copy number of 3 (chromosomes 1,2 and4). When besides these two peaks an extra one appears at value 0.5, some base are also present at the half/half ratio, which indicates a copy number of 4 (like chromosome 3)



Supplemental Figure 4 After aligning CBS1483 S. *pastorianus* reads to the S288C *cerevisiae* reference genome, base calling is performed with Pilon. Plotting the distribution of the Base Calls percentages stored in the .vcf file, seems to confirm Magnolya's copy number prediction. A plot with bell shape centered on the 0.5 value indicates a copy number of two for chromosome 10 (most right figure). A plot with two peaks around 0.33 and 0.66, indicates a copy number of 3 for chromosomes 1, 2 and 4 (first, second and fourth plots). Peaks around 0.33 , 0.66 and 0.5 values suggests a copy number of 4 for chromosome 3 (center figure)

Applying the same principle we can infer the copy numbers from the different clusters in the read count distribution. outputted by PEDCA Supplemental Figure 5. Applied to the first and second clusters of our simulated CBS1483 data set we obtain a typical p=2 shape for the first cluster and p=3 for the second one

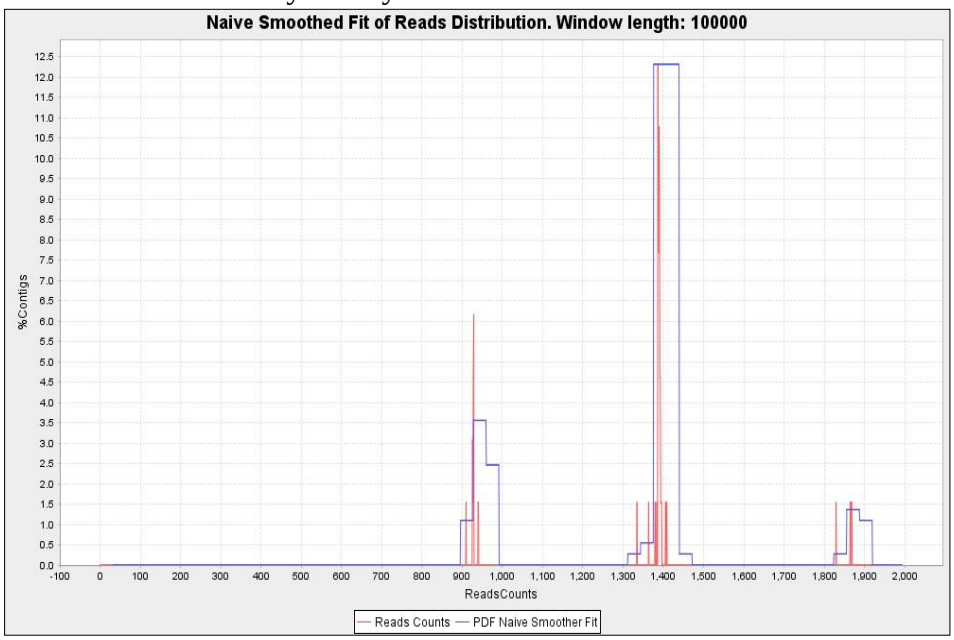


Supplemental Figure 5 Simulated CBS1483 allele feequencies from 1st and 2nd read count distribution clusters.

Results of the naive fit with very large window sizes.

Simulated data

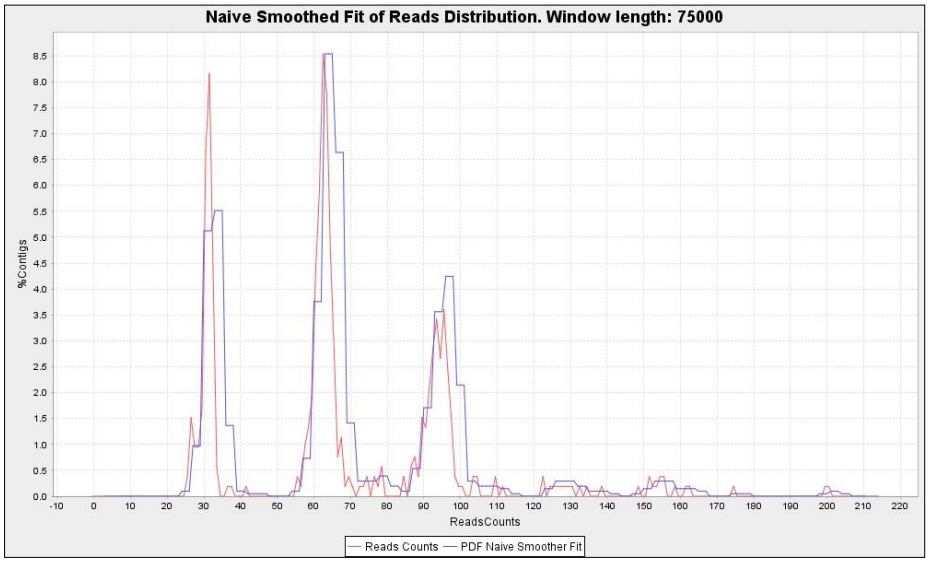
Even with a very large window size (here, up to 100 Kbp) PEDCA fits correctly the main clusters of the read distribution. It is important to take into consideration that the original distribution of the simulated data has very clearly distinct clusters.



Supplemental Figure 6 Naïve smoother fit to Simulated genome with *wl*=100 Kbp

Base Clear Genome

The following plot shows the fit with a *wl*=75 Kbp. *wl*=100Kb is too big for any of the contigs in this genome to be run with PEDCA, so we explored and reached the upper limit of the window size parameter successfully, even with a real dataset.



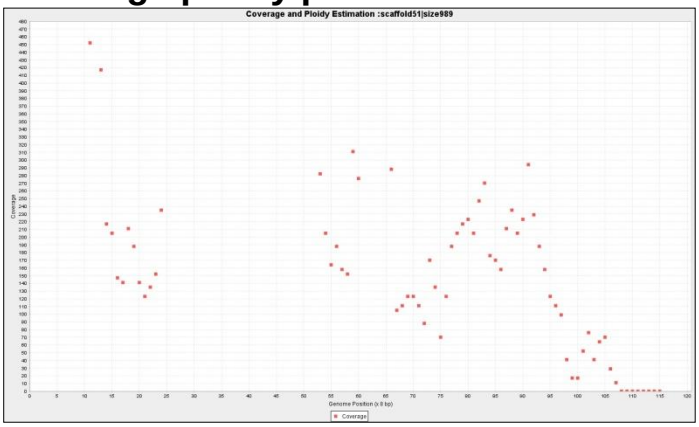
Supplemental Figure 7 Naïve smoother fit to CBS1483 Baseclear genome with *wl*=75 Kbp

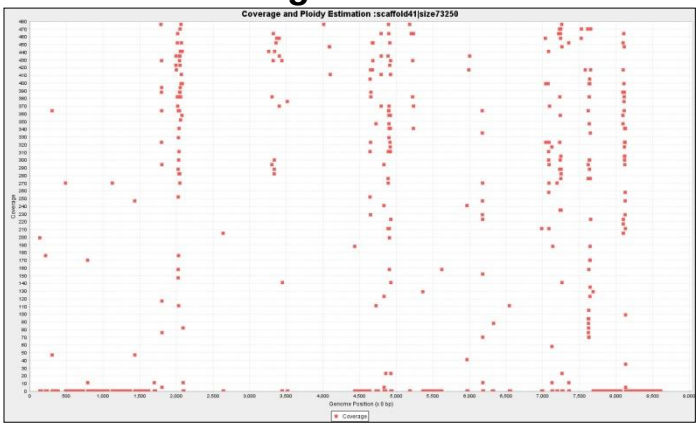
Ploidy estimation fragmentation per round for different window sizes.

	P1	# solved	BC	Total Nb Breaks	1st round	2nd round
40	2	52	1	6769	6706	63
50	1	51	0	1402	1315	85
75	1	51	0	1062	1002	60
100	1	51	0	425	417	8
250	1	51	0	155	147	8
400	1	51	0	77	69	8
500	1	51	0	71	64	7
750	1	51	1	77	63	14
1000	1	51	0	73	54	19
1500	1	51	0	88	50	38
2000	1	51	0	102	48	54
2500	1	51	0	97	46	51
3000	1	51	0	93	47	46
3500	1	51	0	89	35	54
5000	1	51	0	167	44	123
5500	1	51	1	187	41	146
6000	…	…	…	…	…	...
7500	1	51	0	448	39	409
9000	1	51	1	299	37	262
10000	1	51	1	320	35	285
15000	1	51	1	636	30	606
20000	2	51	1	758	27	731
30000	1	51	1	1510	22	1488
40000	1	50	1	4079	15	4064
50000	1	50	1	6028	4	6024
75000	2	47	1	6956	1	6955

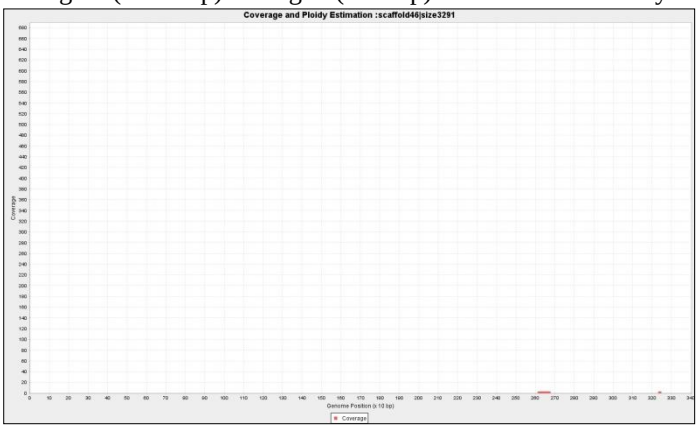
Supplemental Table 4 Ploidy estimation fragmentation per round for different window sizes. Running PEDCA on the Baseclear1483 sequencing shows different amounts of ploidy estimation fragmentation depending of the initial window size. If the initial *wl* is too small, the majority of the contigs will have too many coverage data points with so much noise than the estimation might be compromised. Since the second *wl* will be adapted to the smallest contig unsolved, (and only the small ones will be likely to have been solved) the second run will be better fit to solve the big contigs with less fragmentation. In the other hand a very big *wl* will not solve many of contigs, certainly not the smallest ones, the second *wl* adapted to the smallest unsolved contig which will not contribute much to provide continuous estimations for the rest of the genome. There is a range of ideal initial *wl* in between the two extremes (in this case from 400 bp to 3.500bp).

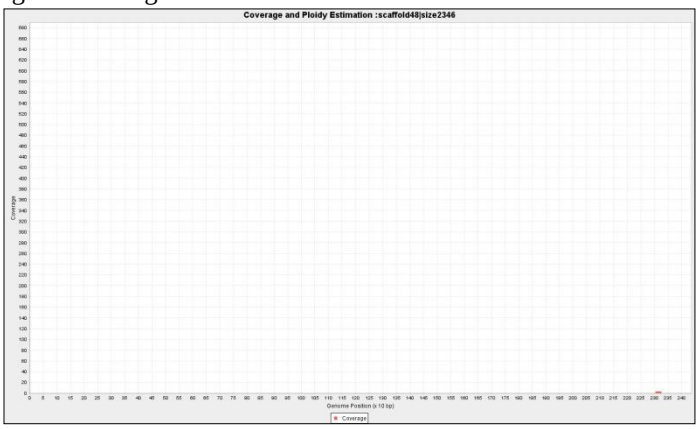
Coverage/ploidy plots for the 8 common unsolved contigs in Baseclear CBS1483



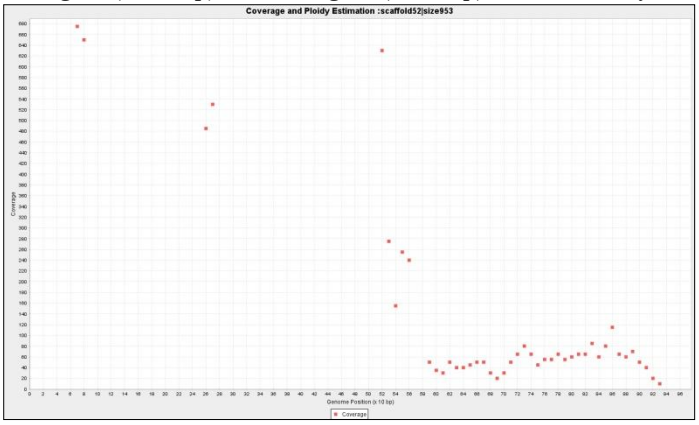


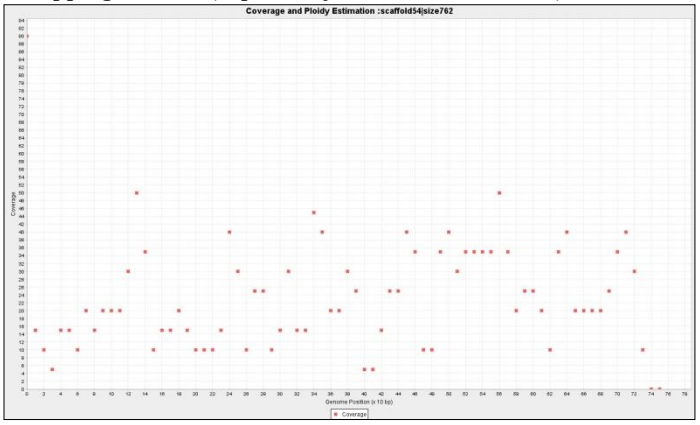
Contig 51 (989 Kbp) Contig41 (73 Kbp) and both have a very irregular coverage information to make a reliable estimation



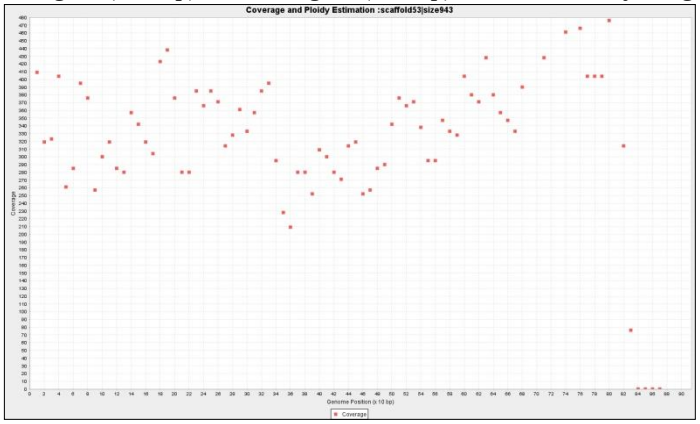


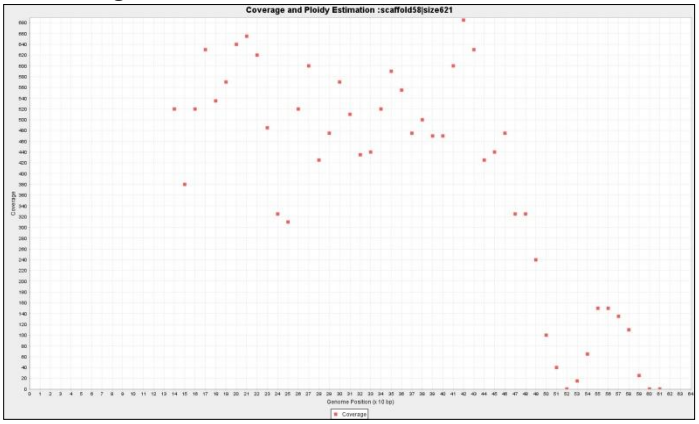
Contig 46 (3291 bp) and Contig 48 (2346 bp) don't have any reads mapping to them (is probably mitochondrial DNA)





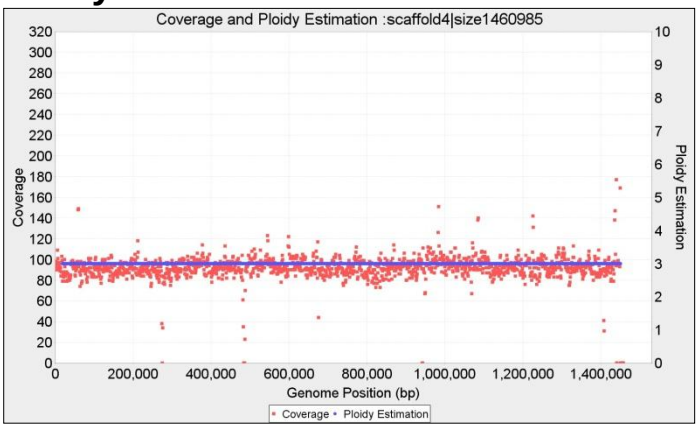
Contig 52 (953 bp) and Contig 54 (762 bp) both have a very irregular coverage information to make a reliable estimation

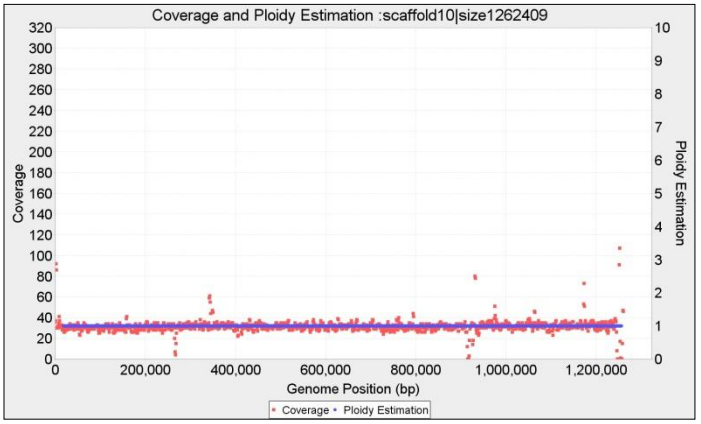




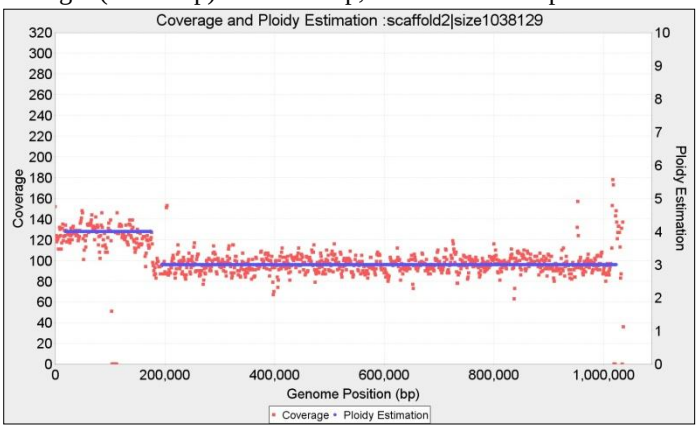
Contig 53 (943 bp) and Contig 58 (621 bp) both have a very irregular coverage information to make a reliable estimation

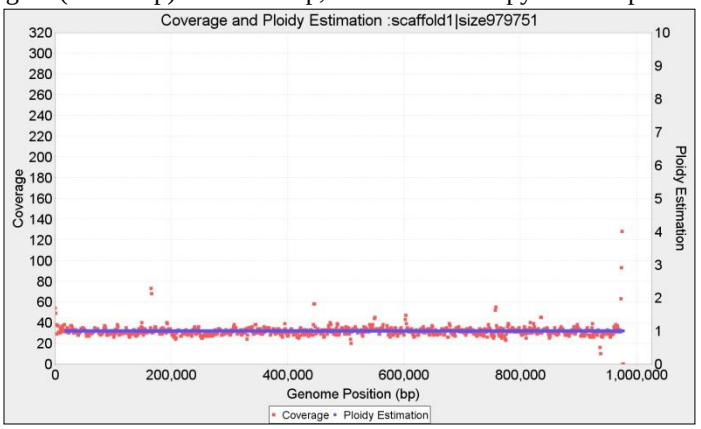
Ploidy estimation for Baseclear CBS1483 from window sizes (2 Kb < *wl* < 16bp)



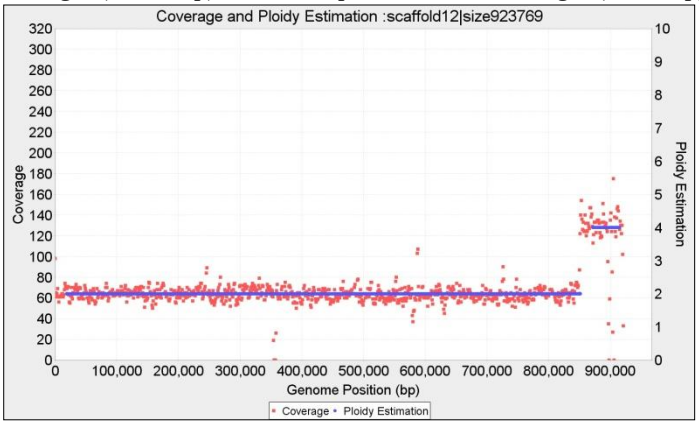


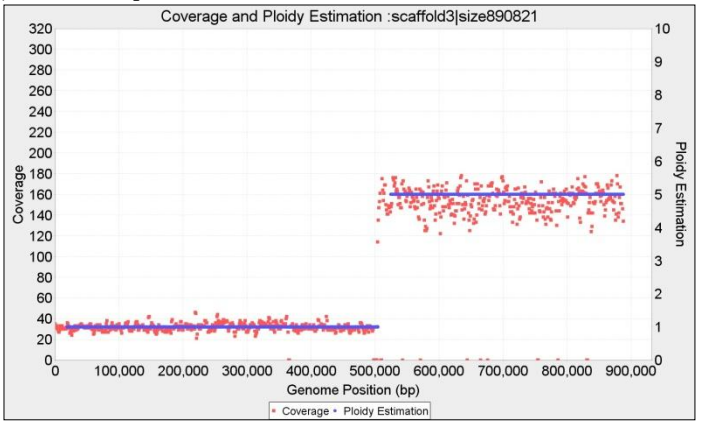
Contig 4 (1460Kbp) wl=2.000 bp, with continuous p=3 and Contig 10 (1262Kbp) wl=2.000 bp, with constant copy number p=1



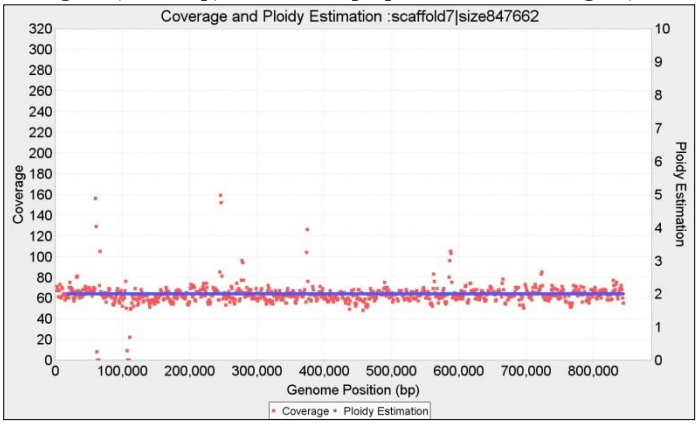


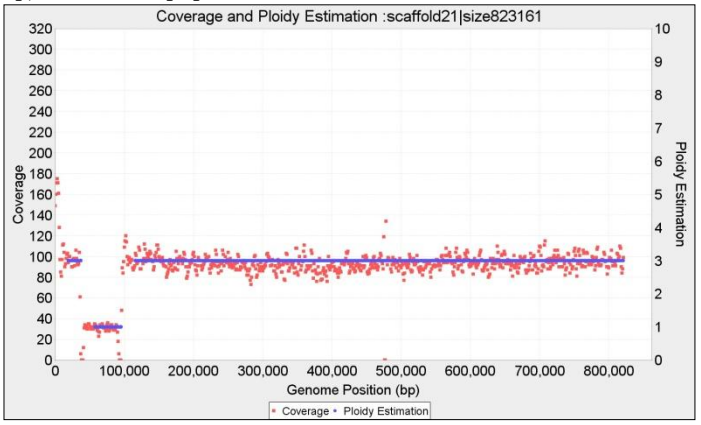
Contig 2 (1038Kbp) wl=2.000 p={4,3,4} and Contig 1 (979 Kbp) wl=2.000 p=1





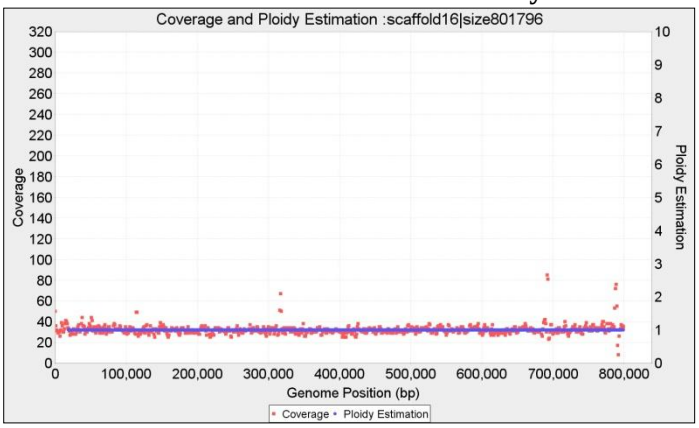
Contig 12 (923 Kbp) wl=2.000 bp, p={2,4} and Contig 3 (890 Kbp) wl=2.000 bp, p={1,5}

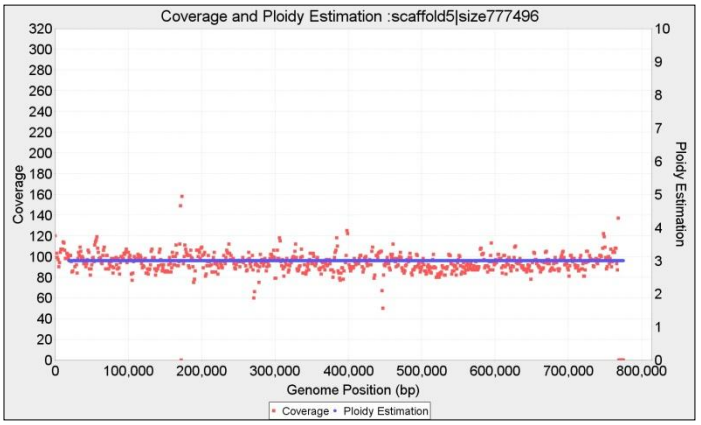




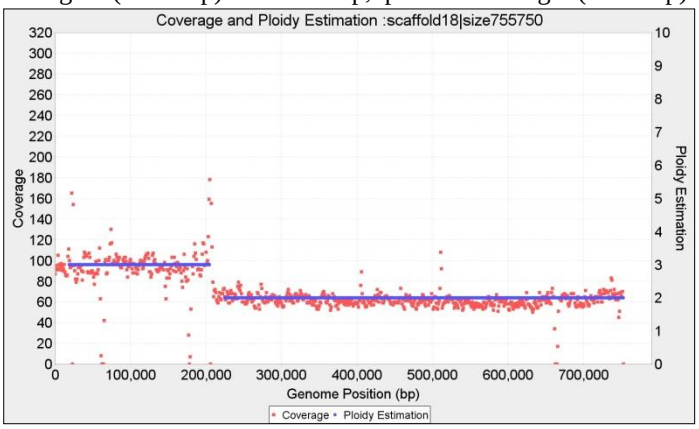
Contig 7 (847 Kbp) wl=2.000 bp, p={2} and Contig 21 (823 Kbp) wl=2.000 bp, with fragmented copy numbers P={3,1,3}

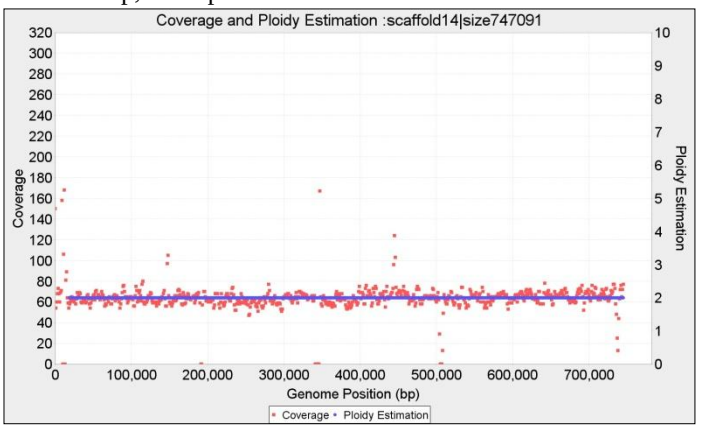
Ploidy estimation for Baseclear CBS1483



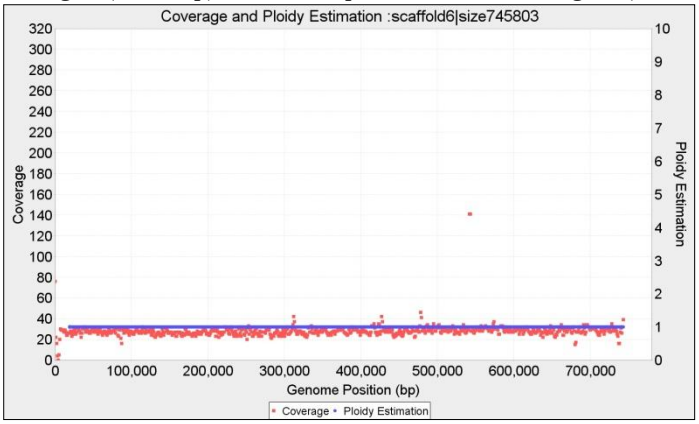


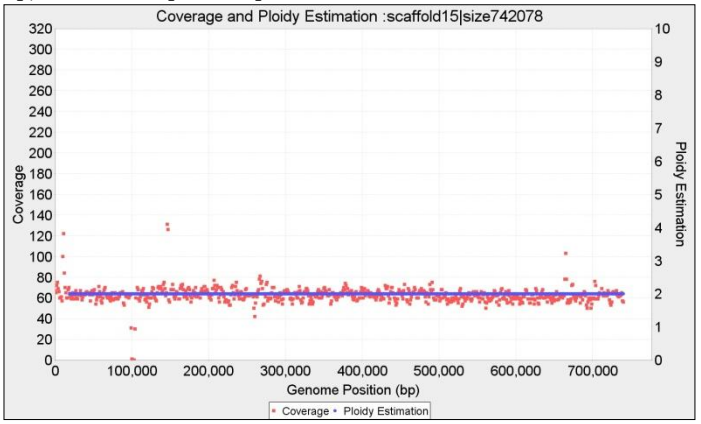
Contig 16 (801 Kbp) wl=2.000 bp, p=1 and Contig 5 (777 Kbp) wl=2.000 bp, with p=3



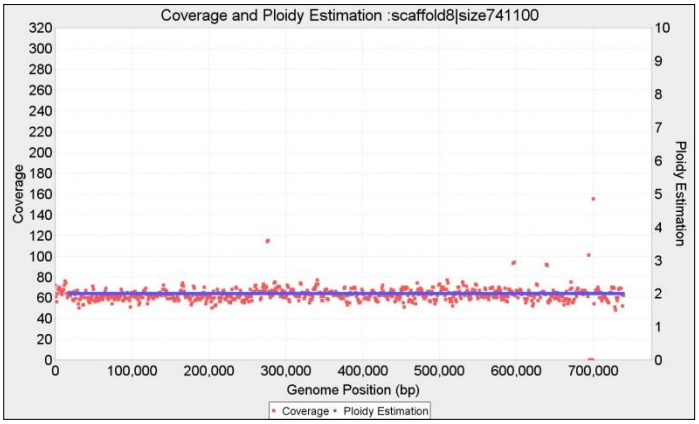


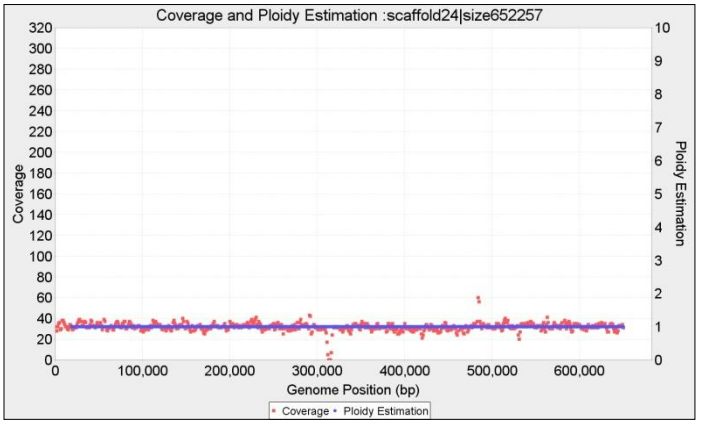
Contig18 (755 Kbp) wl=2.000 bp, P={3,2} and Contig 14 (747 Kbp) wl=2.000 bp, with p=2





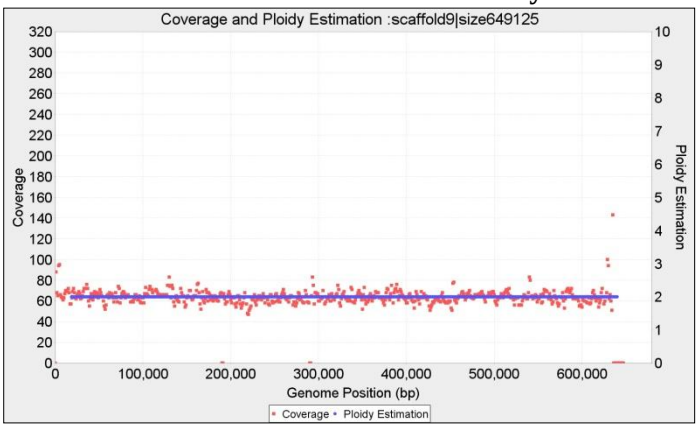
Contig 6 (745 Kbp) p=1 wl=2.000 and Contig 15 (742 Kbp) wl=2.000 with p=2

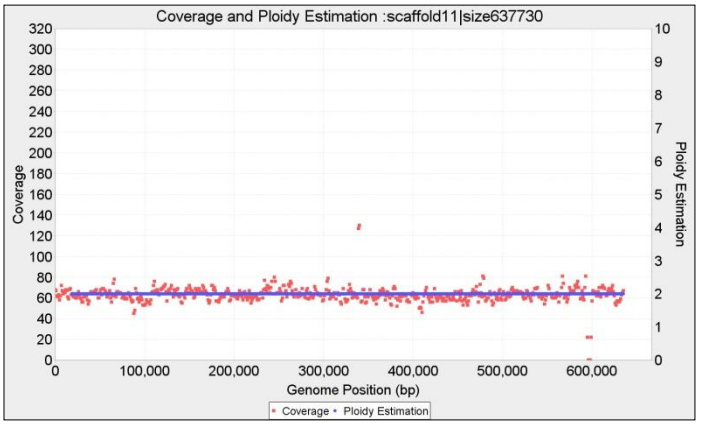




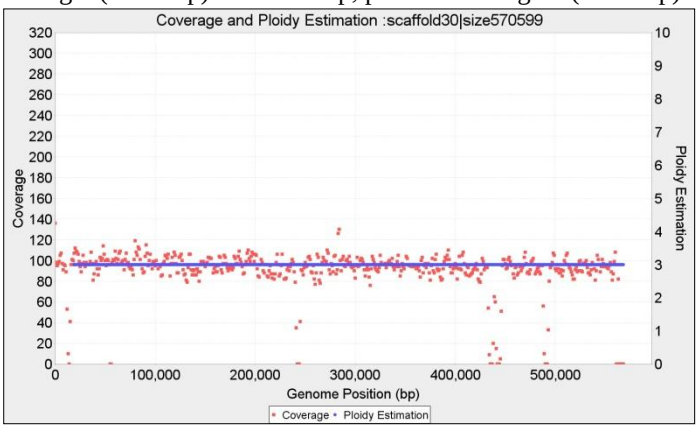
Contig 8 (741 Kbp) wl=2.000 bp, p=2 and Contig 24 (652 Kbp) wl=2.000 bp, with p=1

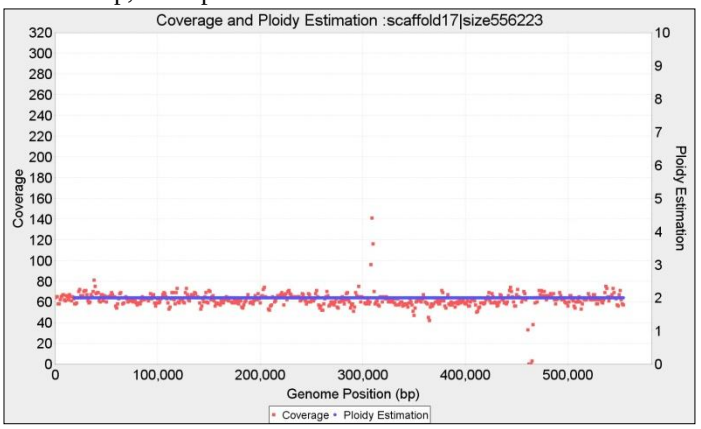
Ploidy estimation for Baseclear CBS1483



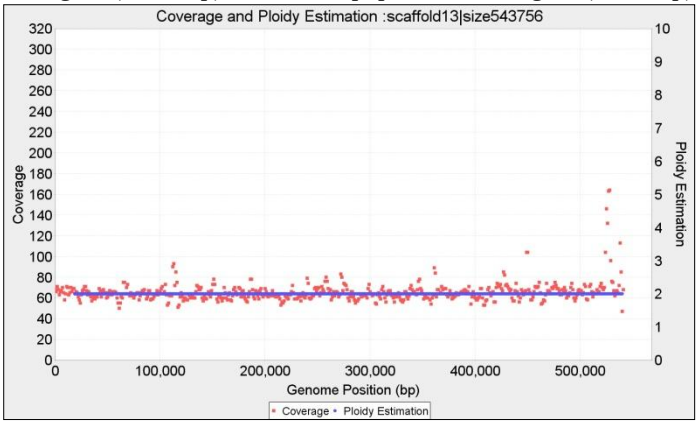


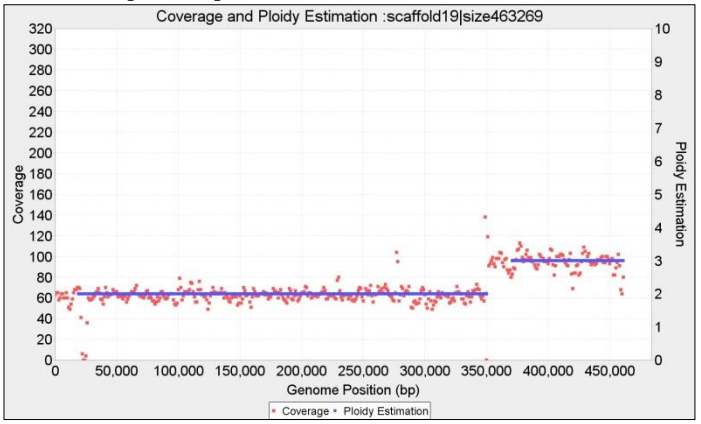
Contig 9 (649 Kbp) wl=2.000 bp, p=2 and Contig 11 (637 Kbp) wl=2.000 bp, with p=2



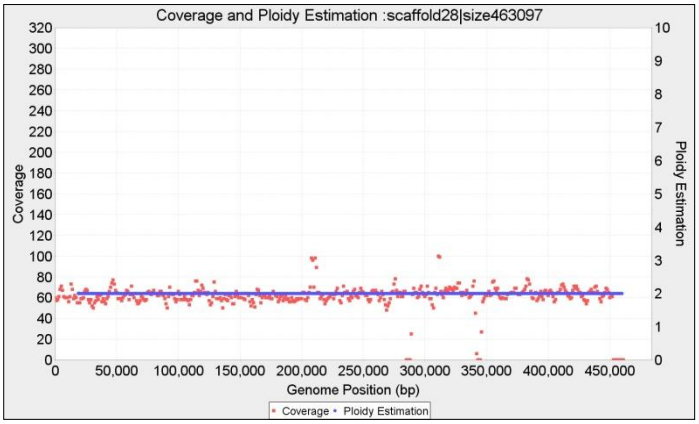


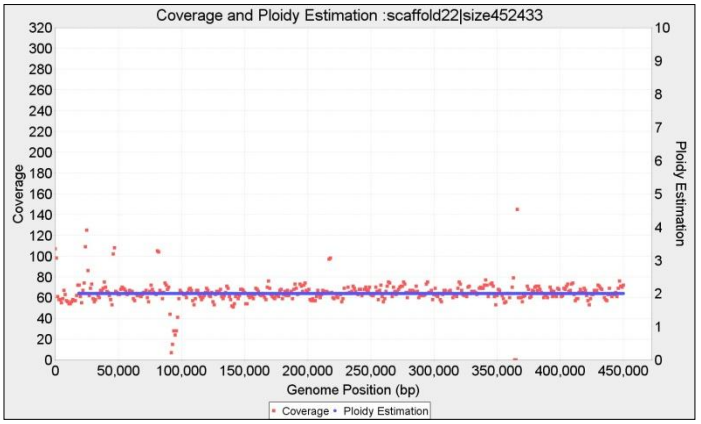
Contig 30 (570 Kbp) wl=2.000 bp, p=3 and Contig 17 (556 Kbp) wl=2.000 bp, with p=2





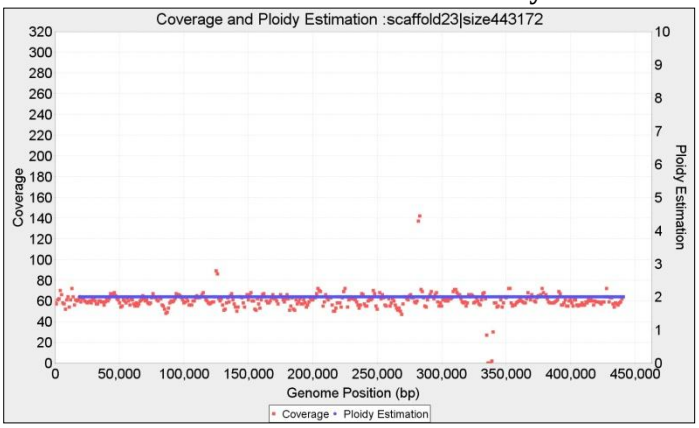
Contig 13 (543 Kbp) wl=2.000 bp, p=2 and Contig 19 (463 Kbp) wl=2.000 bp, with P={2,3}

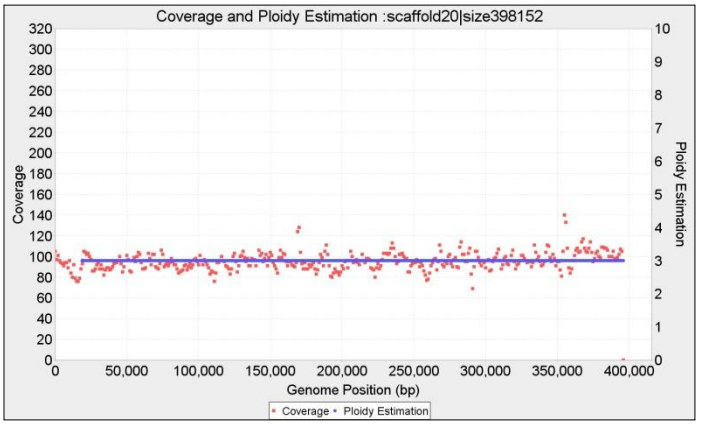




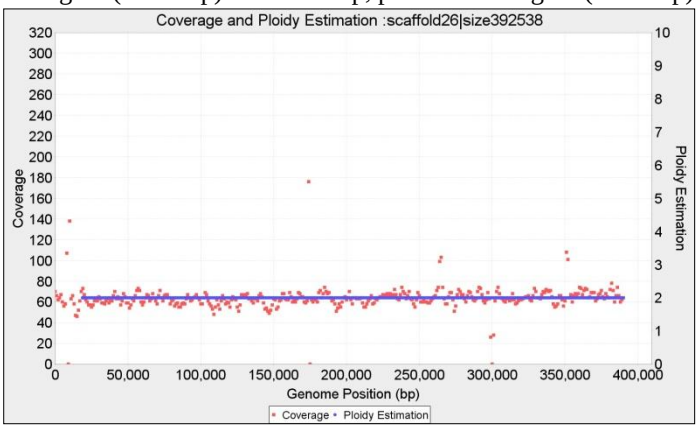
Contig 28 (463 Kbp) wl=2.000 bp, p=2 and Contig 22 (452 Kbp) wl=2.000 bp, with p=2

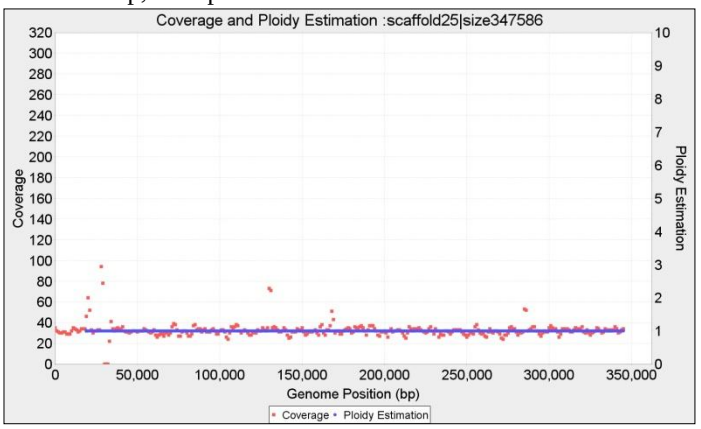
Ploidy estimation for Baseclear CBS1483



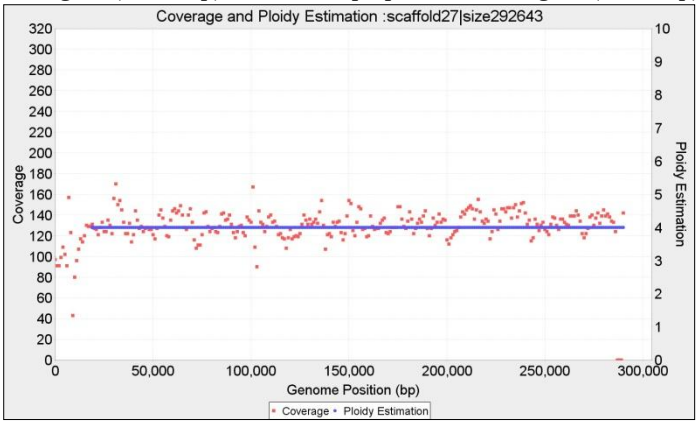


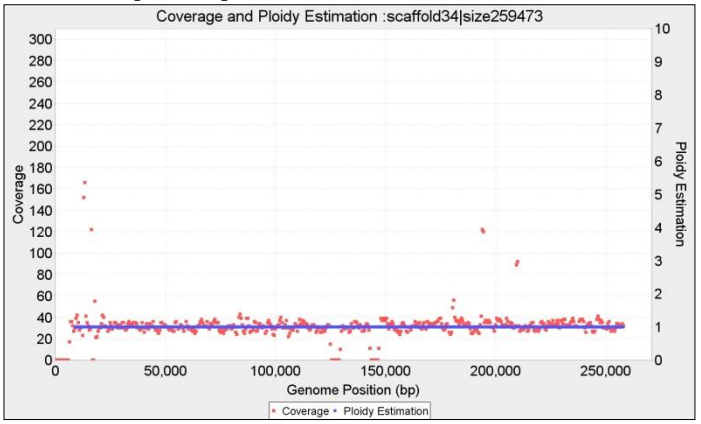
Contig 23 (443 Kbp) wl=2.000 bp, p=2 and Contig 20 (398 Kbp) wl=2.000 bp, with p=3



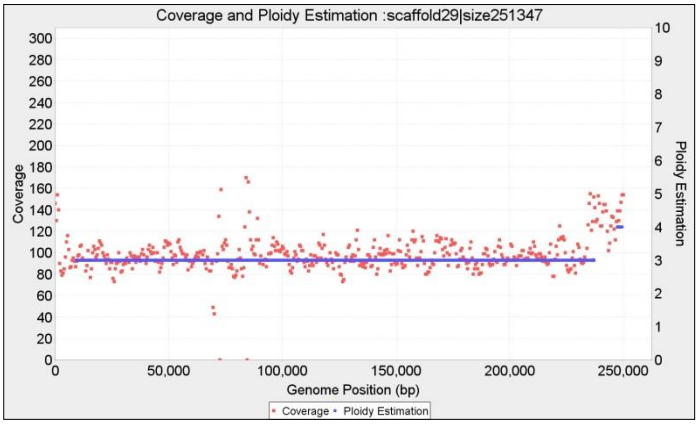


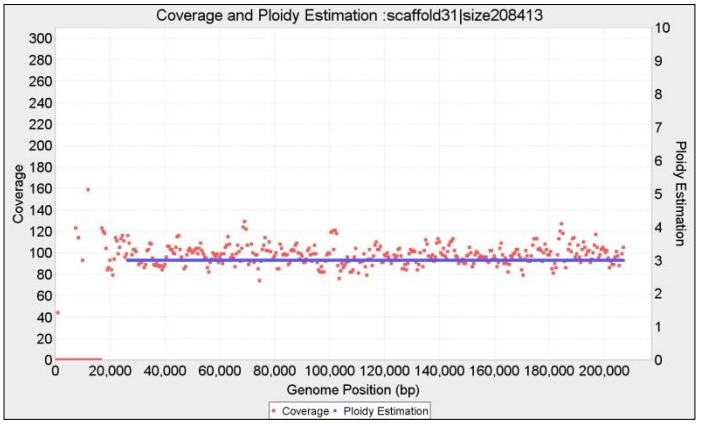
Contig 26 (463 Kbp) wl=2.000 bp, p=2 and Contig 25 (398 Kbp) wl=2.000 bp, with p=1





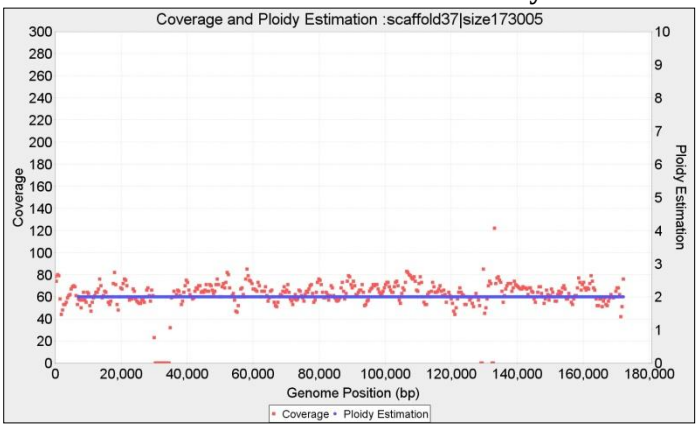
Contig 27 (292 Kbp) wl=2.000 bp, P={4} and Contig 34 (252 Kbp) wl=1.000 bp, with p=1

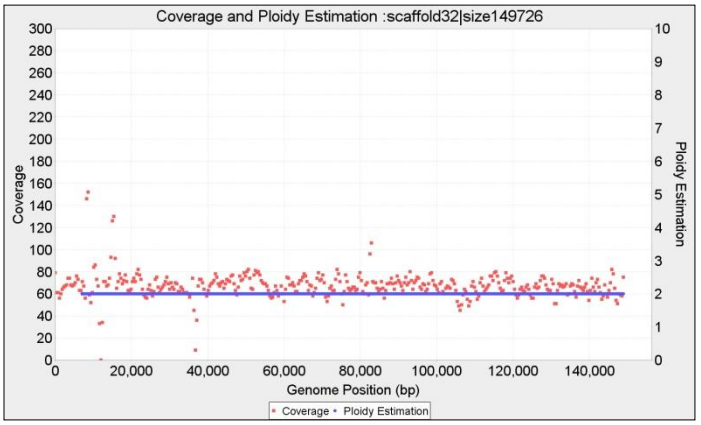




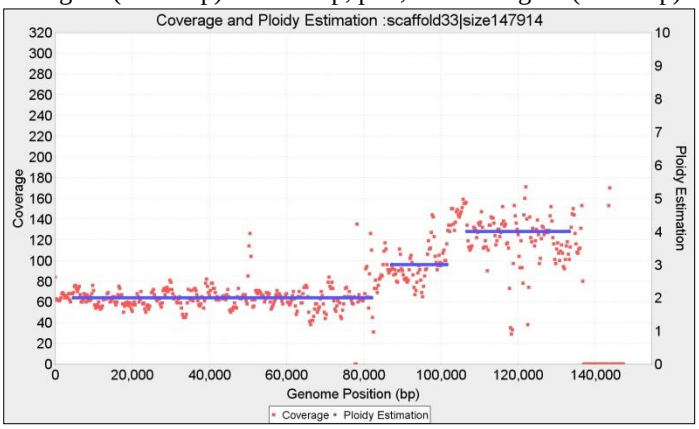
Contig 29 (251 Kbp) wl=1.000 bp, P={3,4} and Contig 31 (208 Kbp) wl=1.000 bp, with p=3

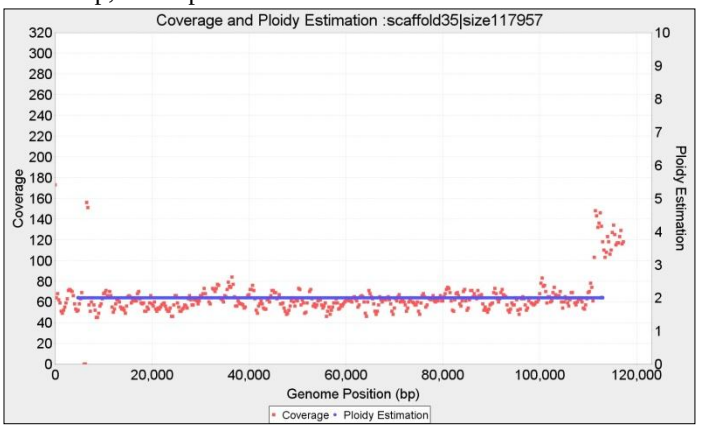
Ploidy estimation for Baseclear CBS1483



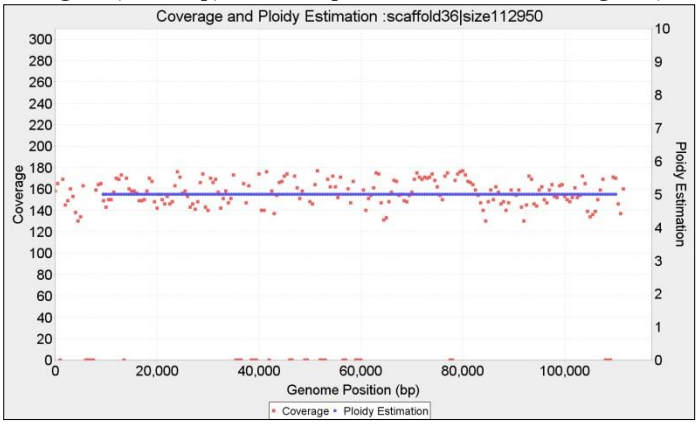


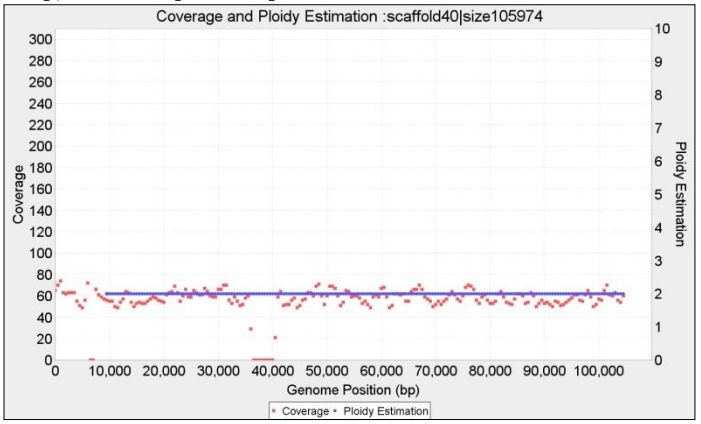
Contig 37 (173 Kbp) wl=750 bp, p=2; and Contig 32 (149 Kbp) wl=750 bp, with p=2



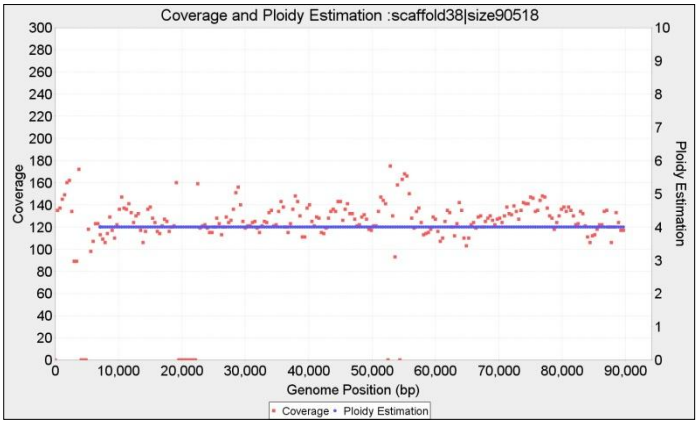


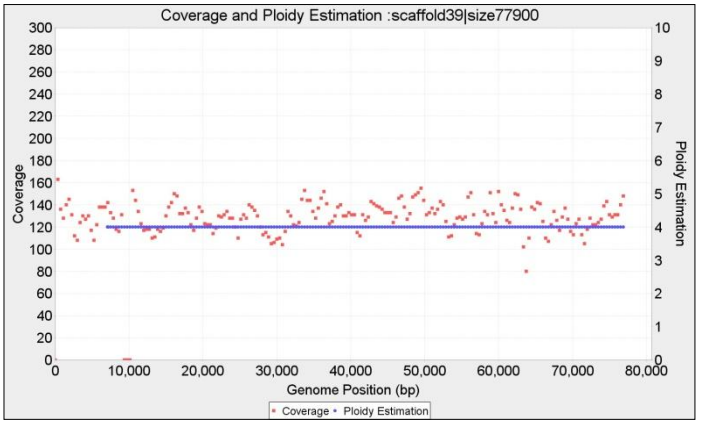
Contig 33 (147 Kbp) wl=500 bp, P={2,3,4} and Contig 35 (117 Kbp) wl=500 bp, with p=2





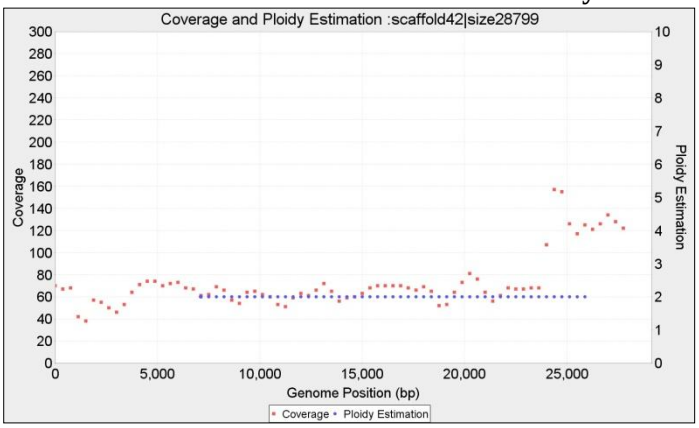
Contig 36 (112 Kbp) wl=1.000 bp, p=5 and Contig 40 (105 Kbp) wl=1.000 bp, with p=2

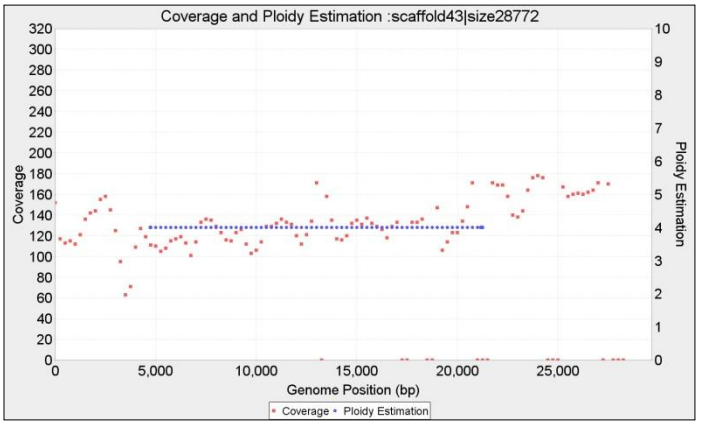




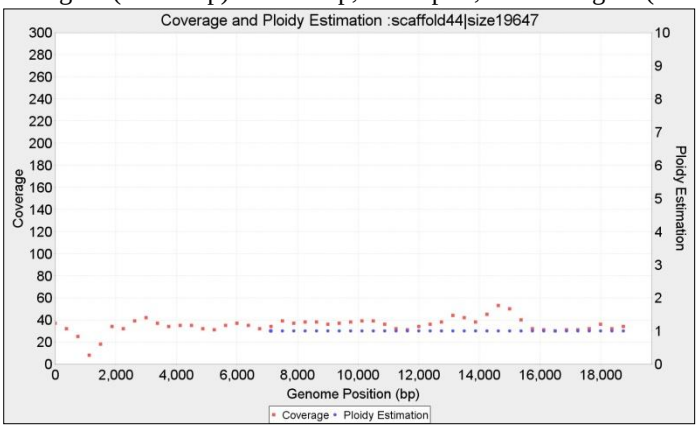
Contig 38 (90 Kbp) wl=750 bp, p=4 and Contig 39 (77 Kbp) wl=750 bp, with p=4,

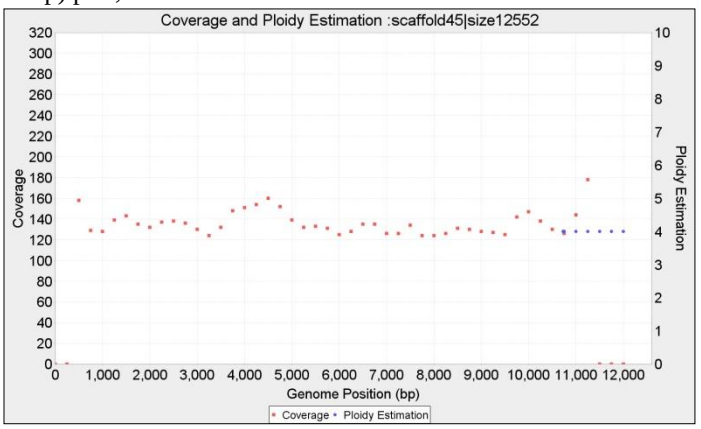
Ploidy estimation for Baseclear CBS1483



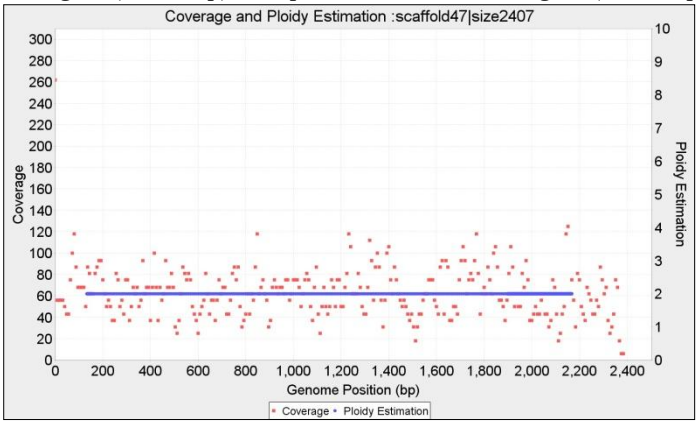


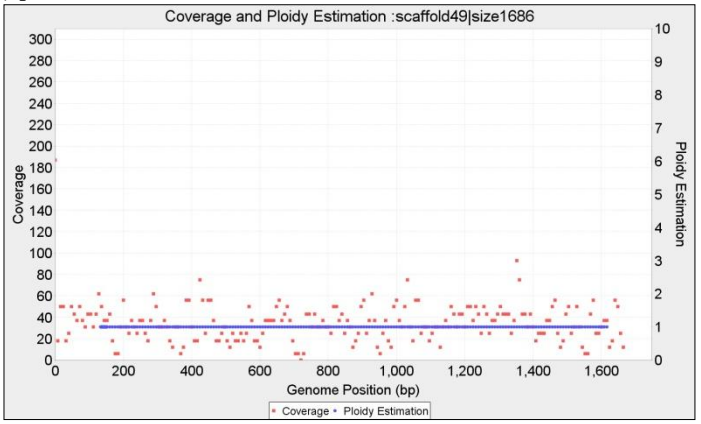
Contig 42 (28.7 Kbp) wl=750 bp, with p=2; and Contig 43 (28.7 Kbp) p=4, wl=500



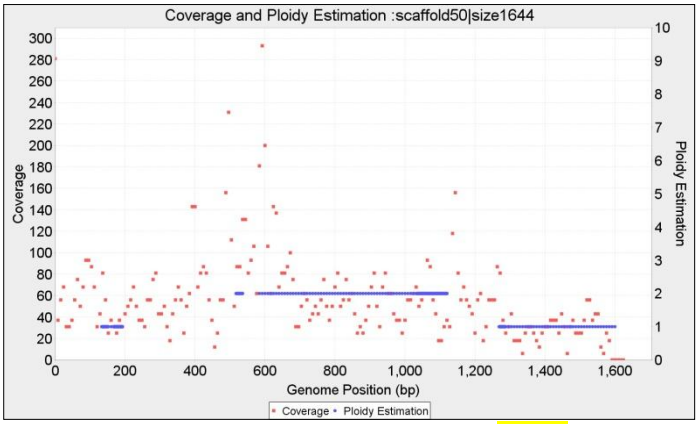


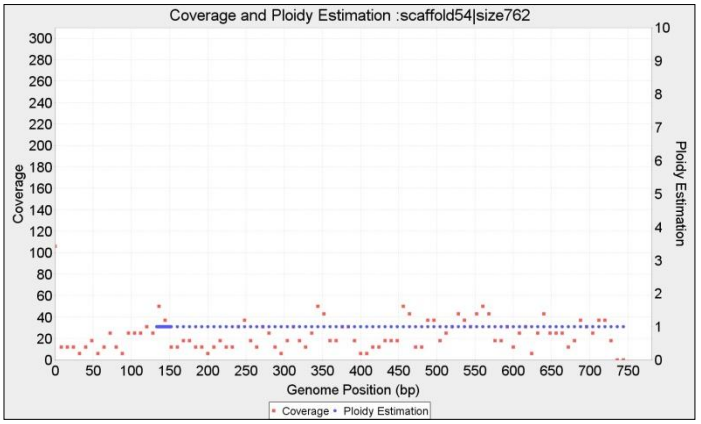
Contig 44 (19.6 Kbp) with p=1, wl=400 and Contig 45 (12.5 Kbp) p=4, wl=400





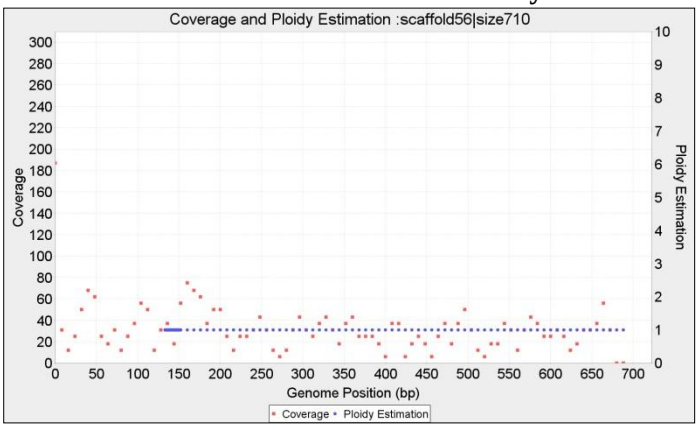
Contig 47 (2.407 bp) wl=20 with p=2, and Contig 49 (1.686 bp) wl=20, p=1,

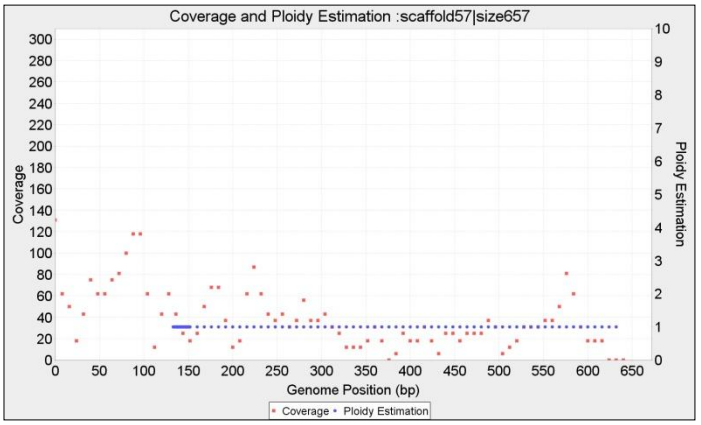




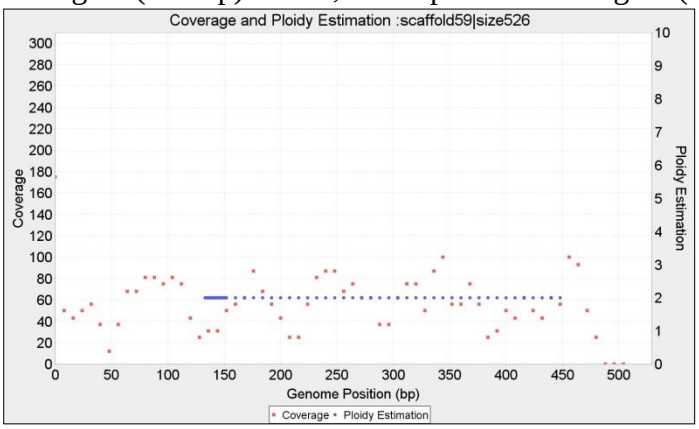
Contig 50 (1.644 bp) *wl*=20, with P={2?,1?} and Contig 54 (762 bp) *wl*=20, p=1

Ploidy estimation for Baseclear CBS1483



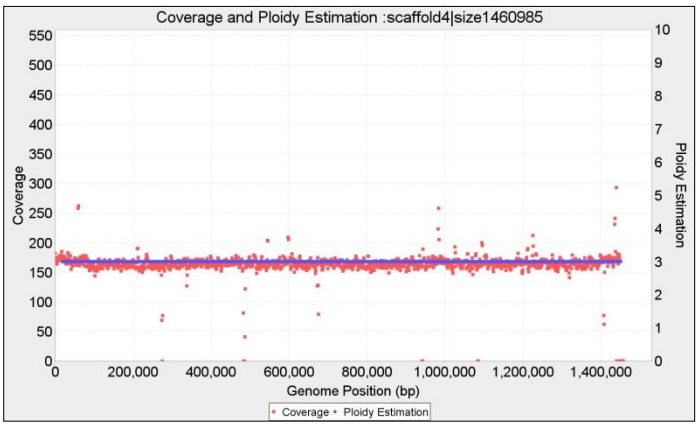


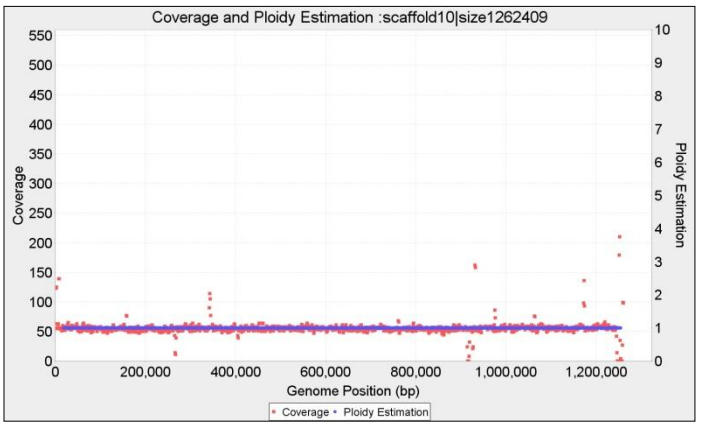
Contig 56 (710 bp) *wl*=16, with p=1 and Contig 57 (657 bp) *wl*=16, p=1



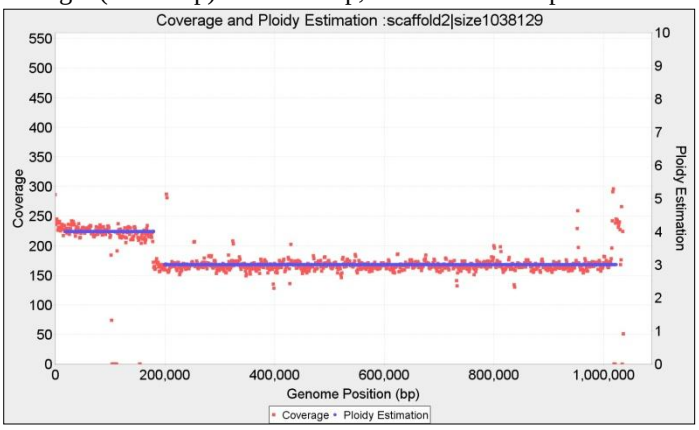
Contig 59 (526 bp) *wl*=16, p=2

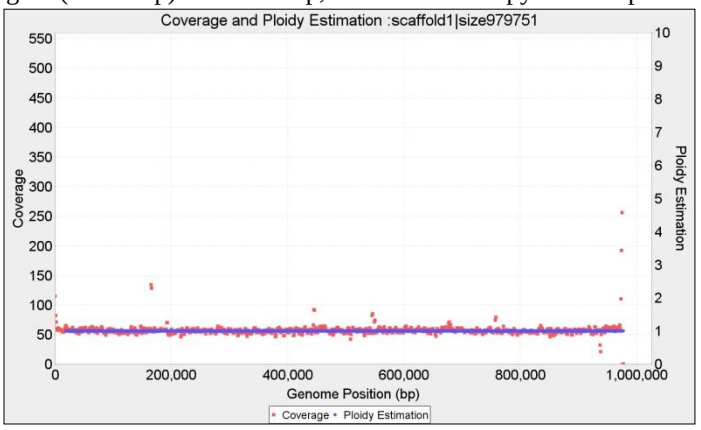
Ploidy estimation for Novogene CBS1483 from selected *wl's* (2 Kb- 16bp)



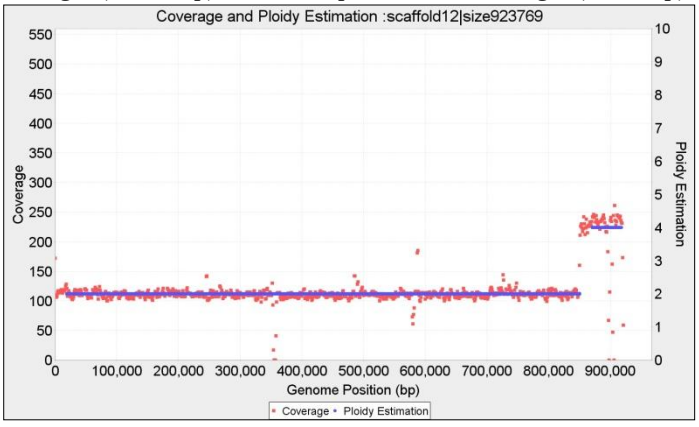


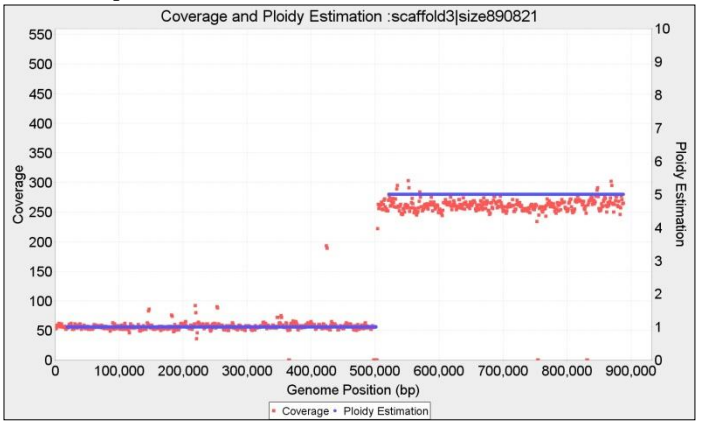
Contig 4 (1460Kbp) wl=2.000 bp, with continuous p=3 and Contig 10 (1262Kbp) wl=2.000 bp, with constant copy number p=1



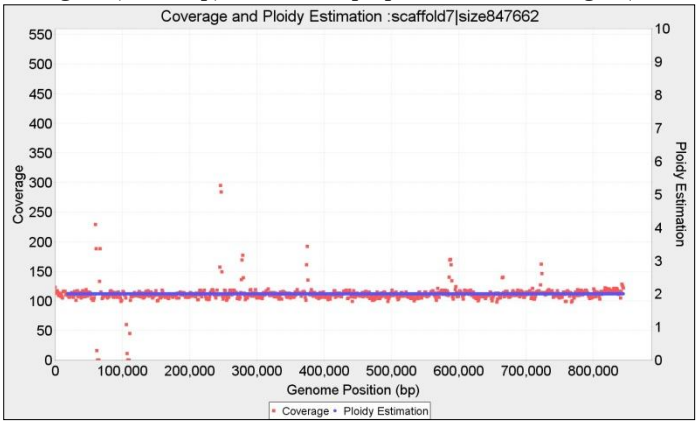


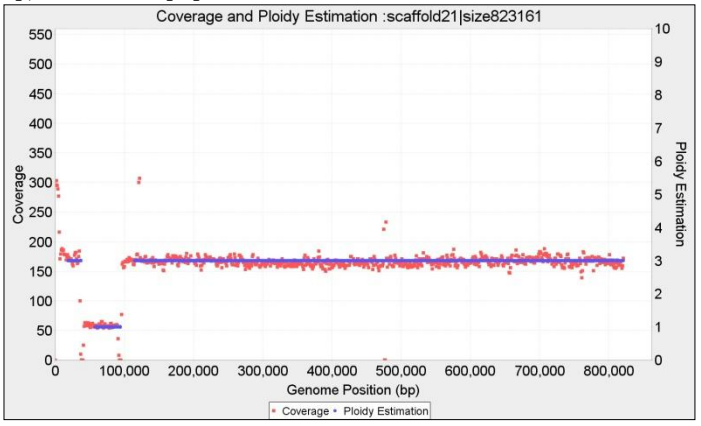
Contig 2 (1038Kbp) wl=2.000 p={4,3} and Contig 1 (979 Kbp) wl=2.000 p=1





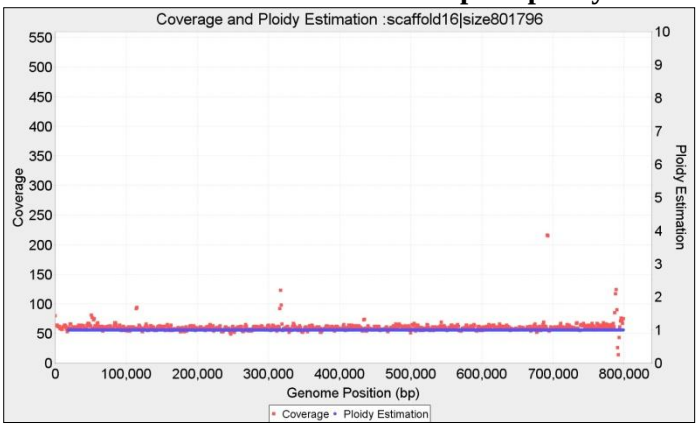
Contig 12 (923 Kbp) wl=2.000 bp, p={2,4} and Contig 3 (890 Kbp) wl=2.000 bp, p={1,5}

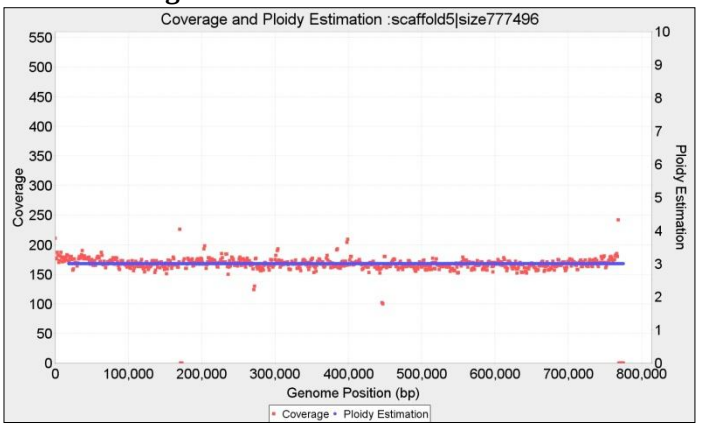




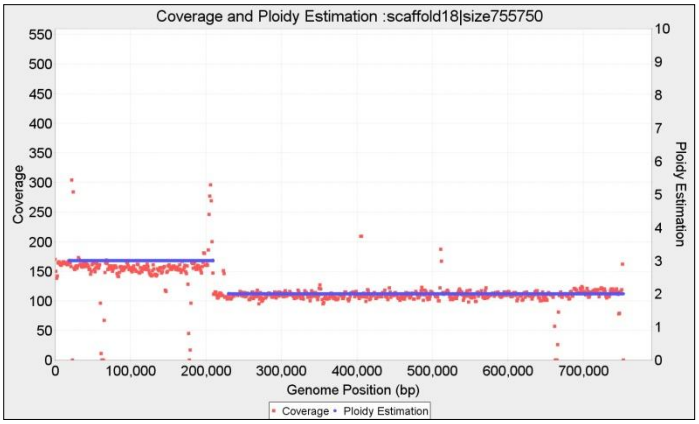
Contig 7 (847 Kbp) wl=2.000 bp, p={2} and Contig 21 (823 Kbp) wl=2.000 bp, with fragmented copy numbers P={3,1,3}

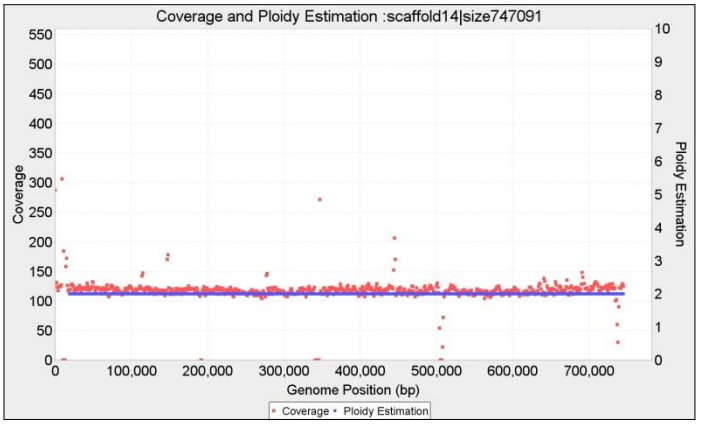
Complete ploidy estimation for Novogene CBS1483



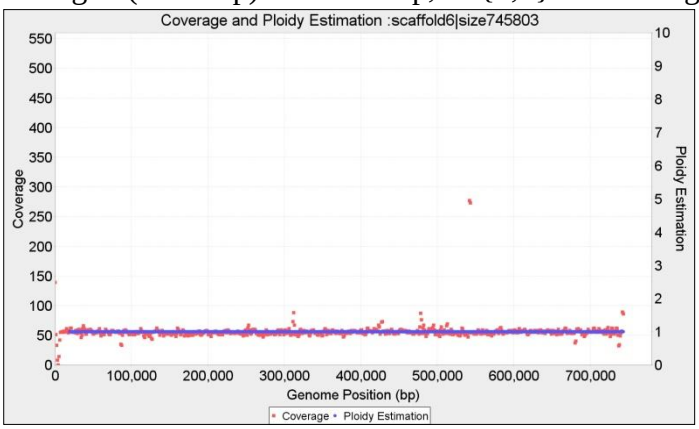


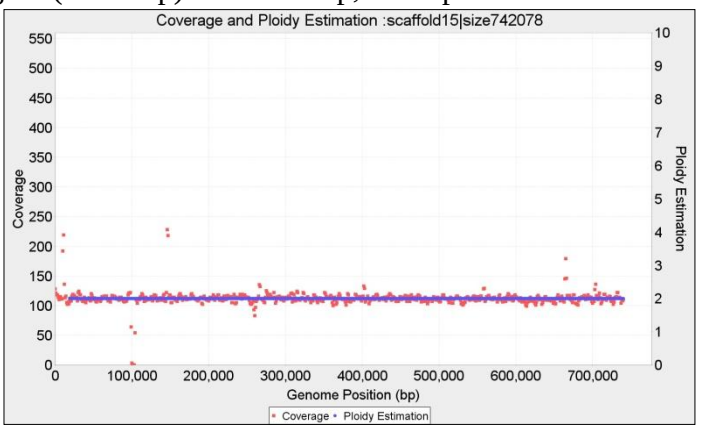
Contig 16 (801 Kbp) *wl*=2.000 bp, p=1 and Contig 5 (777 Kbp) *wl*=2.000 bp, with p=3



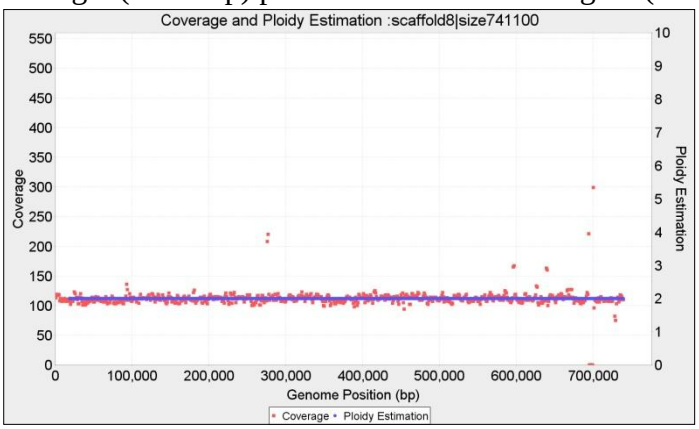


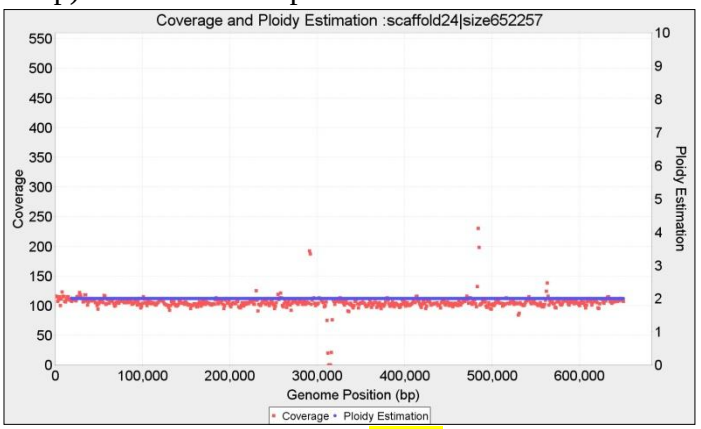
Contig18 (755 Kbp) *wl*=2.000 bp, P={3,2} and Contig 14 (747 Kbp) *wl*=2.000 bp, with p=2





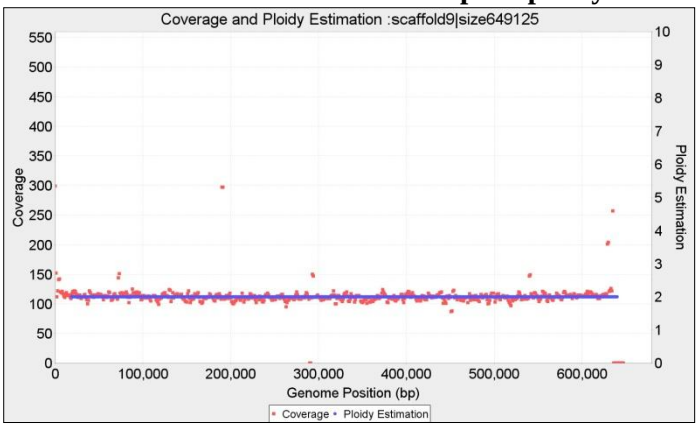
Contig 6 (745 Kbp) p=1 *wl*=2.000 and Contig 15 (742 Kbp) *wl*=2.000 with p=2

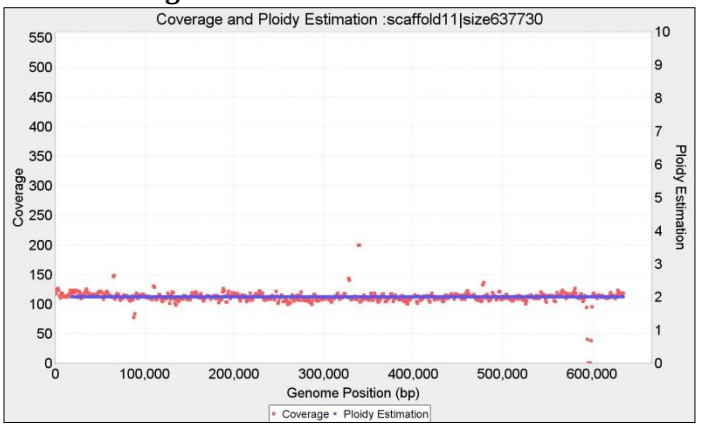




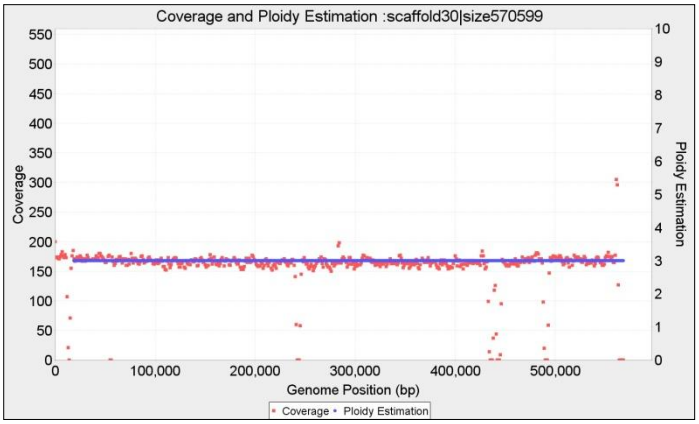
Contig 8 (741 Kbp) *wl*=2.000 bp, p=2 and Contig 24 (652 Kbp) *wl*=2.000 bp, with **p=2!!**

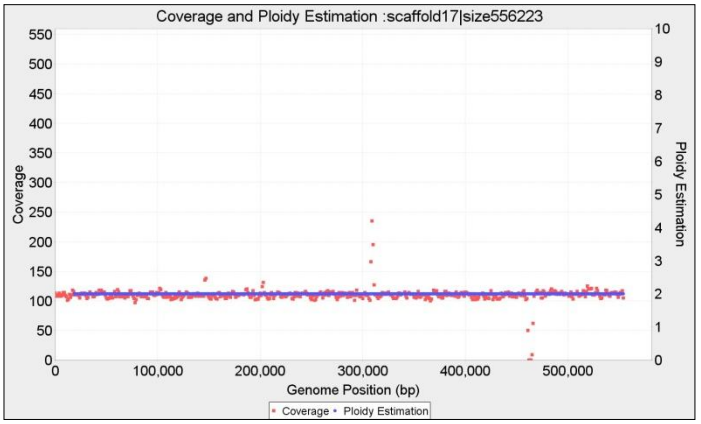
Complete ploidy estimation for Novogene CBS1483



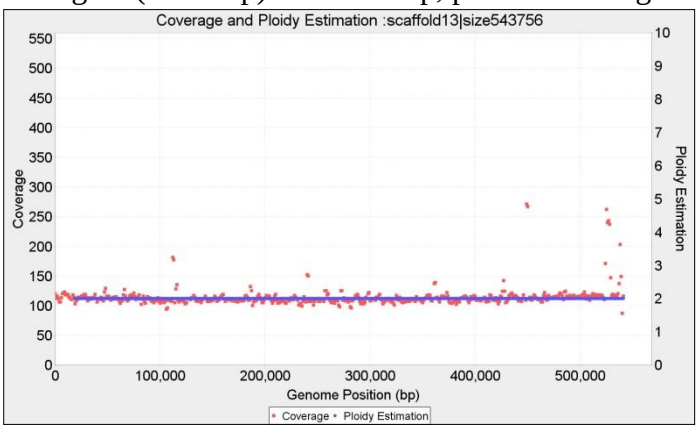


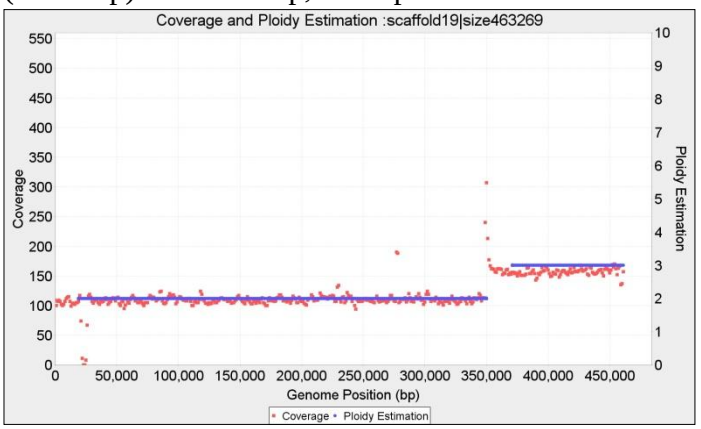
Contig 9 (649 Kbp) *wl*=2.000 bp, p=2 and Contig 11 (637 Kbp) *wl*=2.000 bp, with p=2



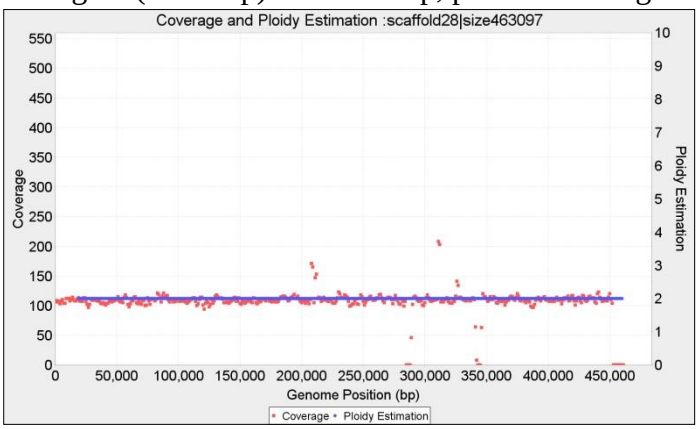


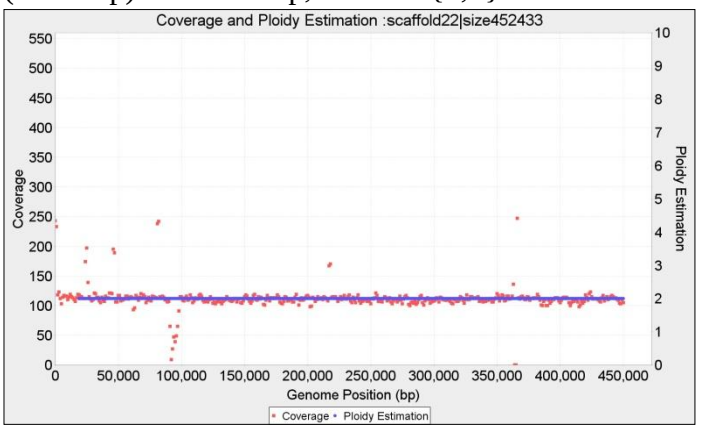
Contig 30 (570 Kbp) *wl*=2.000 bp, p=3 and Contig 17 (556 Kbp) *wl*=2.000 bp, with p=2





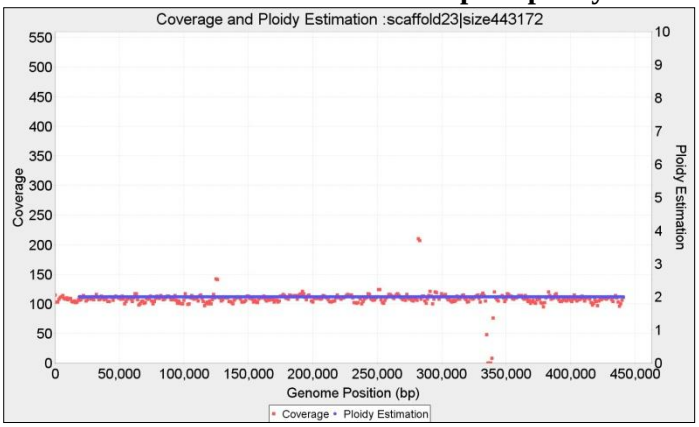
Contig 13 (543 Kbp) *wl*=2.000 bp, p=2 and Contig 19 (463 Kbp) *wl*=2.000 bp, with P={2,3}

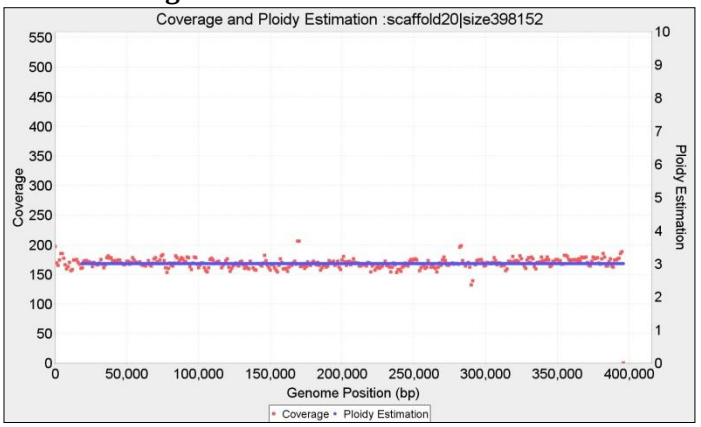




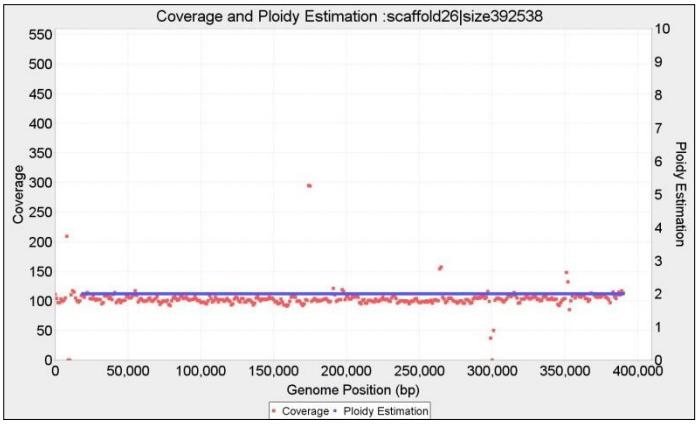
Contig 28 (463 Kbp) *wl*=2.000 bp, p=2 and Contig 22 (452 Kbp) *wl*=2.000 bp, with p=2

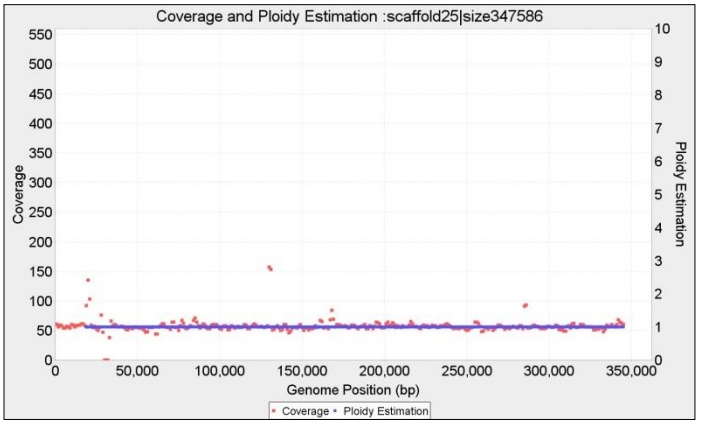
Complete ploidy estimation for Novogene CBS1483



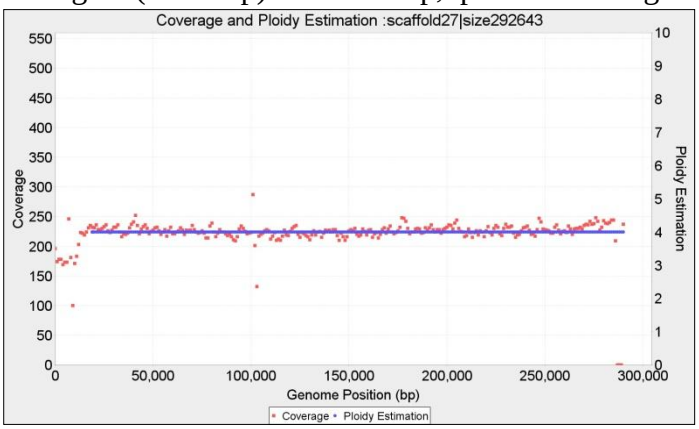


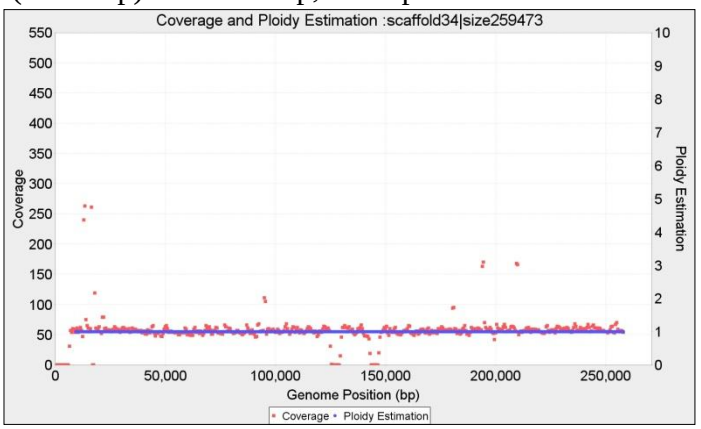
Contig 23 (443 Kbp) *wl*=2.000 bp, p=2 and Contig 20 (398 Kbp) *wl*=2.000 bp, with p=3



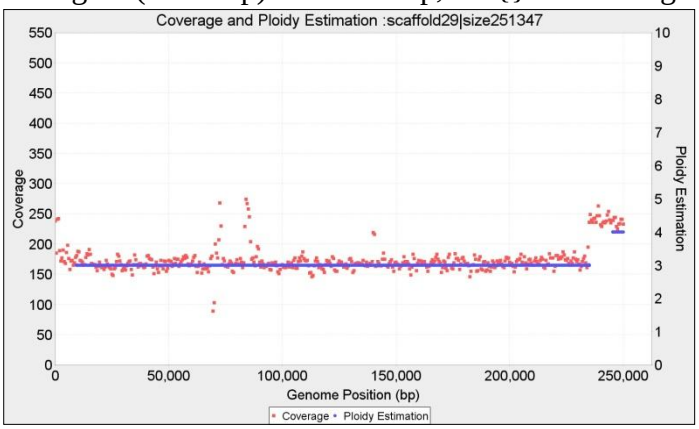


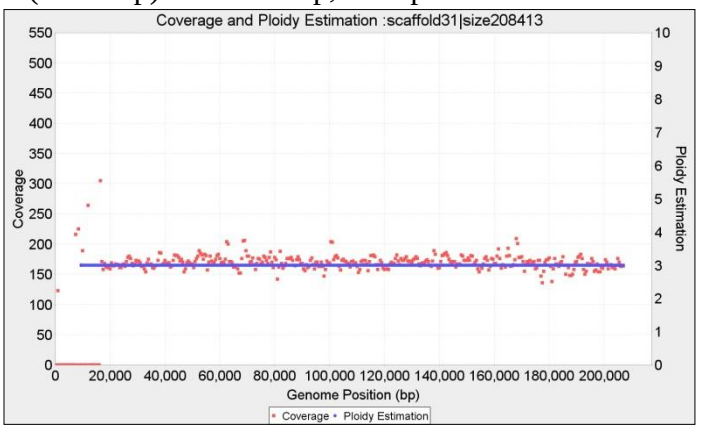
Contig 26 (463 Kbp) *wl*=2.000 bp, p=2 and Contig 25 (398 Kbp) *wl*=2.000 bp, with p=1





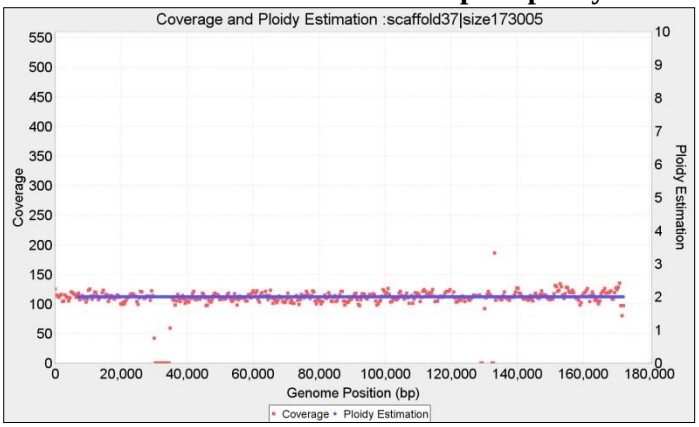
Contig 27 (292 Kbp) *wl*=2.000 bp, P={} and Contig 34 (252 Kbp) *wl*=1.000 bp, with p=1

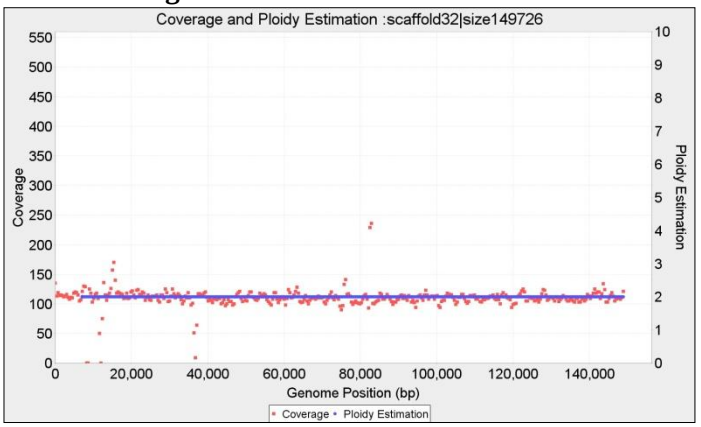




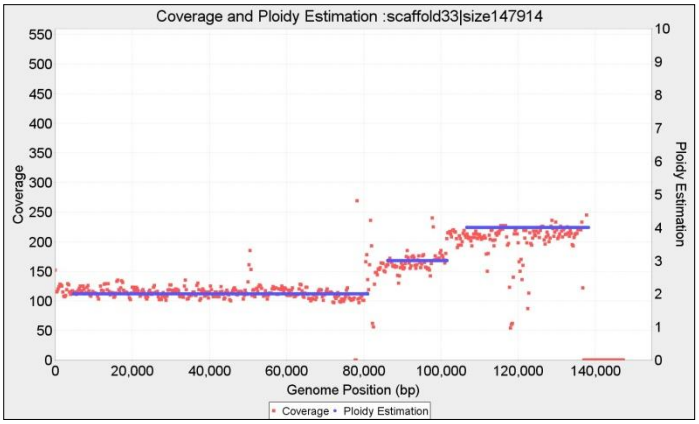
Contig 29 (251 Kbp) *wl*=1.000 bp, P={3,4} and Contig 31 (208 Kbp) *wl*=1.000 bp, with p=3

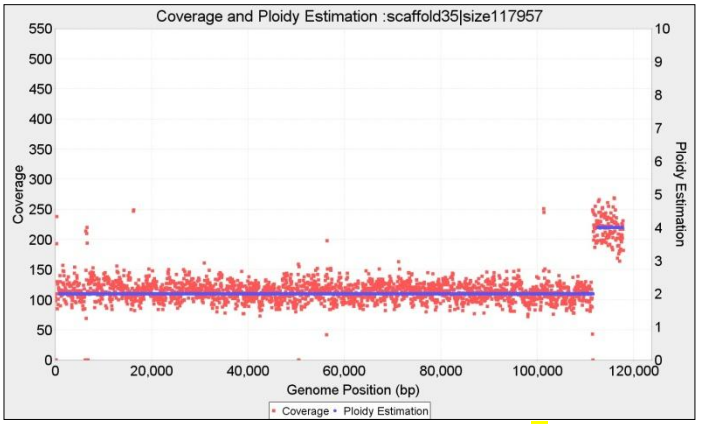
Complete ploidy estimation for Novogene CBS1483



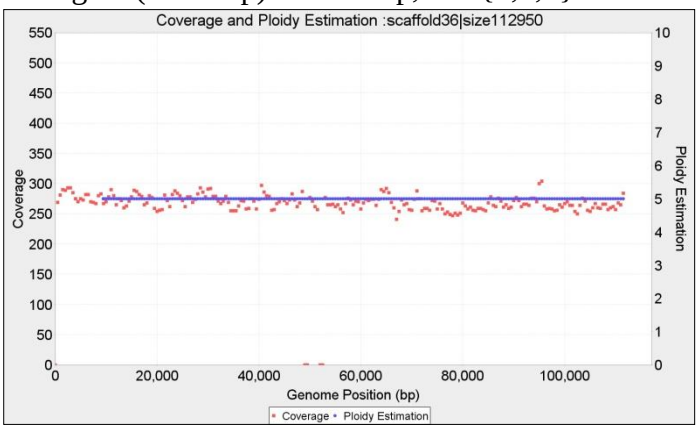


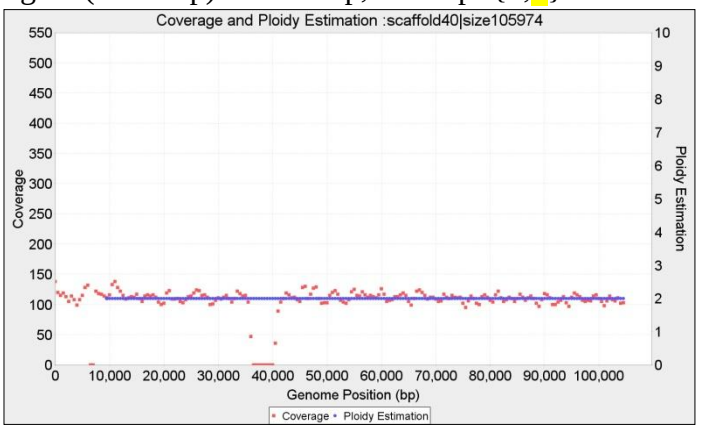
Contig 37 (173 Kbp) *wl*=750 bp, p=2; and Contig 32 (149 Kbp) *wl*=750 bp, with p=2



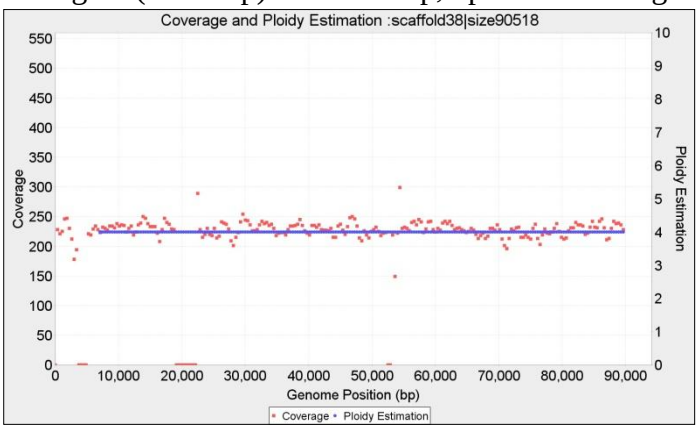


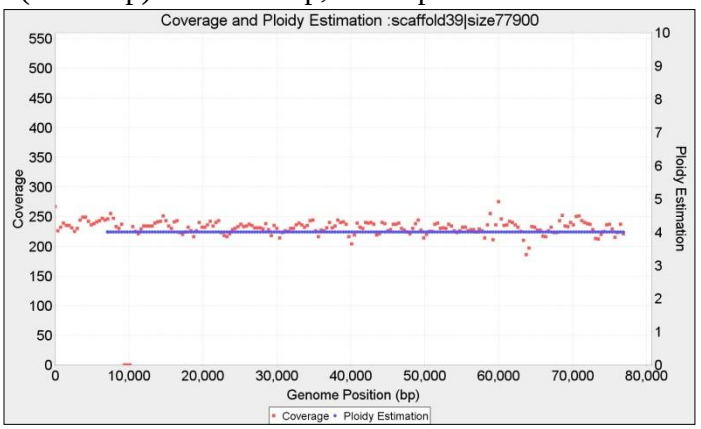
Contig 33 (147 Kbp) *wl*=500 bp, P={2,3,4} and Contig 35 (117 Kbp) *wl*=100 bp, with p={2,4}





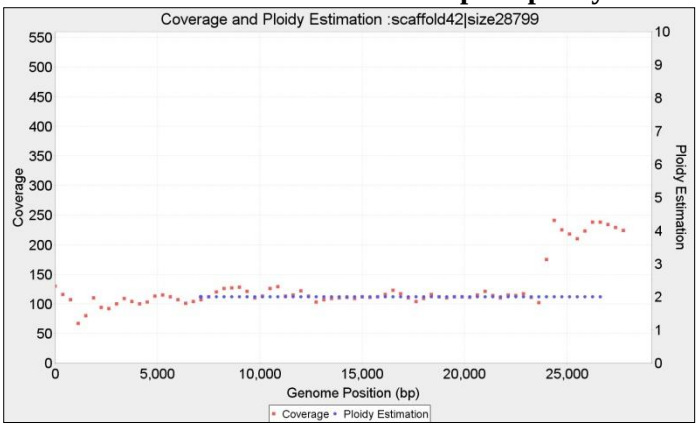
Contig 36 (112 Kbp) *wl*=1.000 bp, p=5 and Contig 40 (105 Kbp) *wl*=1.000 bp, with p=2

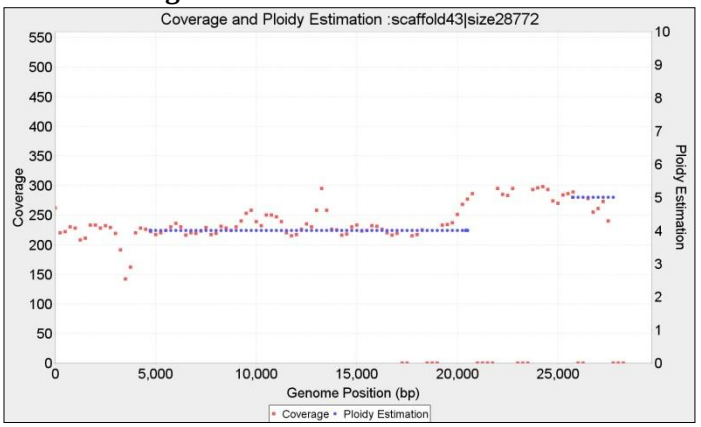




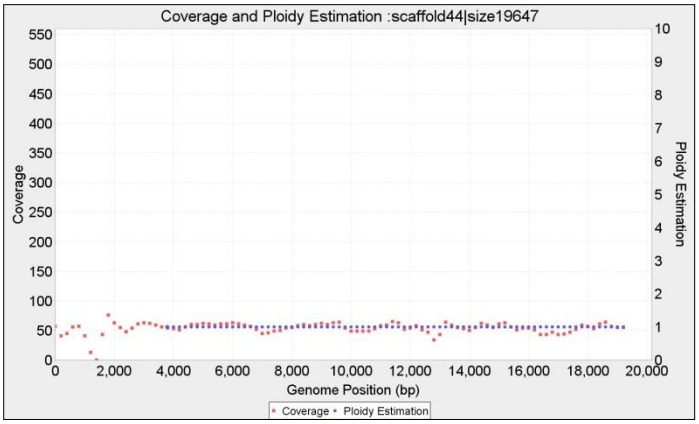
Contig 38 (90 Kbp) *wl*=750 bp, p=4 and Contig 39 (77 Kbp) *wl*=750 bp, with p=4,

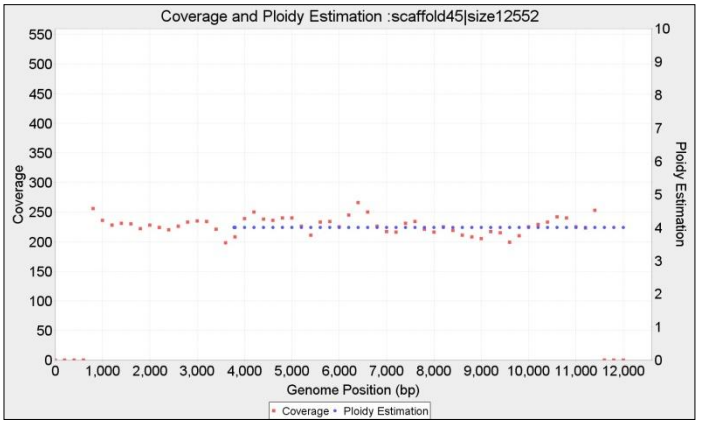
Complete ploidy estimation for Novogene CBS1483



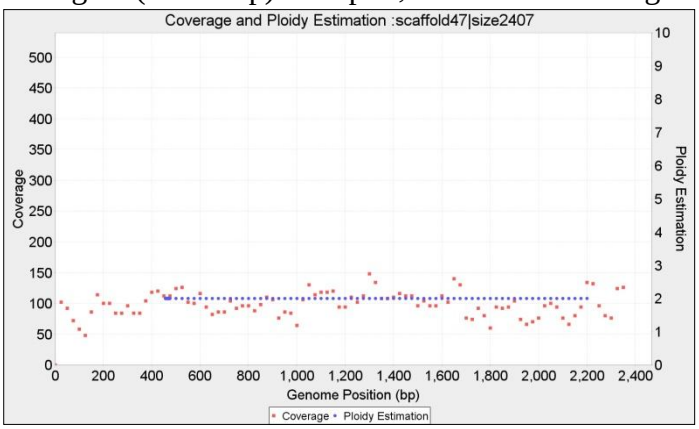


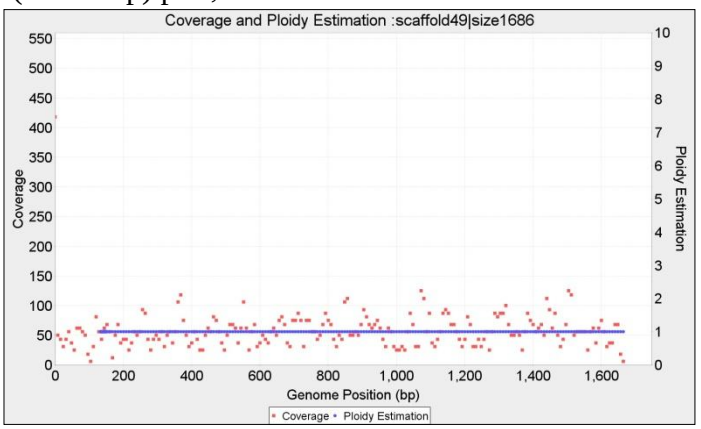
Contig 42 (28.7 Kbp) *wl*=750 bp, with p=2; and Contig 43 (28.7 Kbp) p=4, *wl*=500



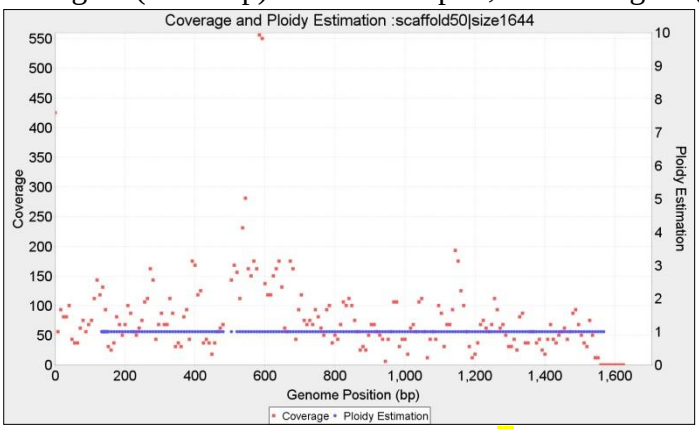


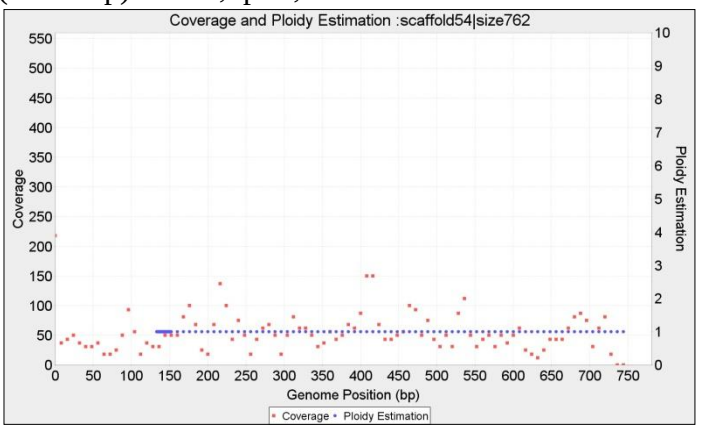
Contig 44 (19.6 Kbp) with p=1, *wl*=400 and Contig 45 (12.5 Kbp) p=4, *wl*=400





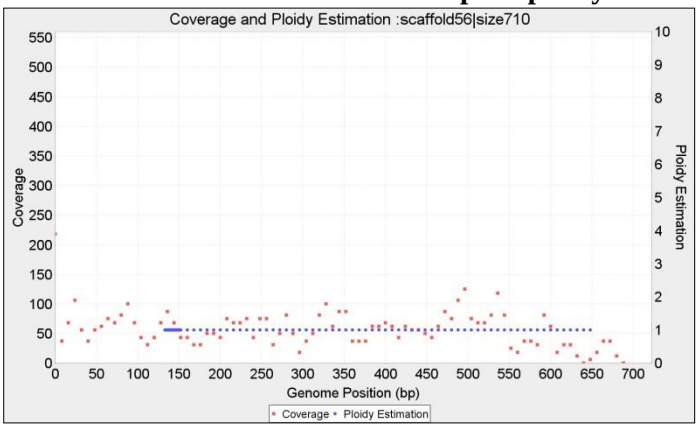
Contig 47 (2.407 bp) *wl*=50 with p=2, and Contig 49 (1.686 bp) *wl*=20, p=1,

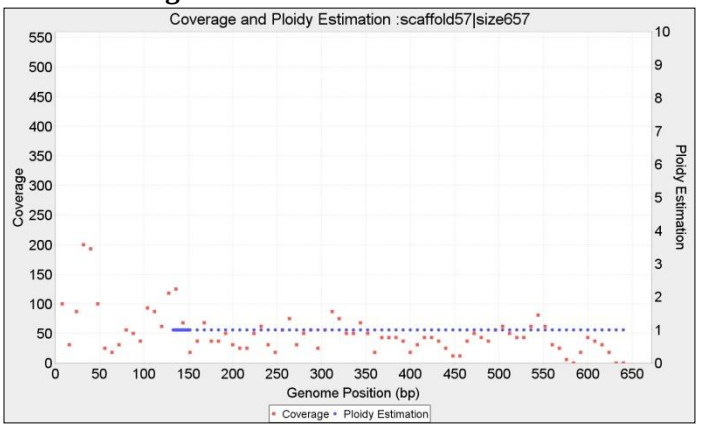




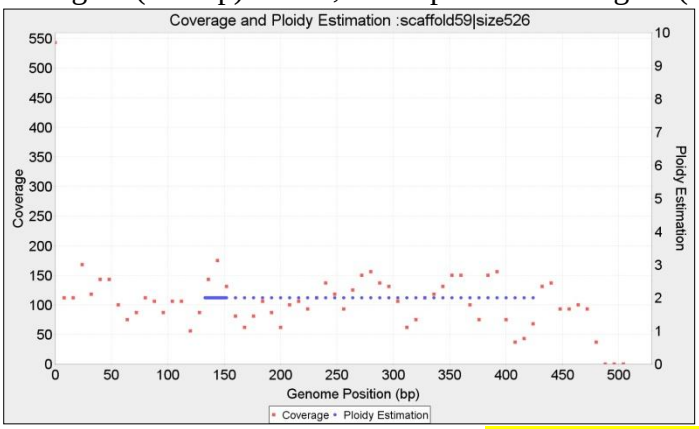
Contig 50 (1.644 bp) *wl*=20, with P={1} and Contig 54 (762 bp) *wl*=20, p=1

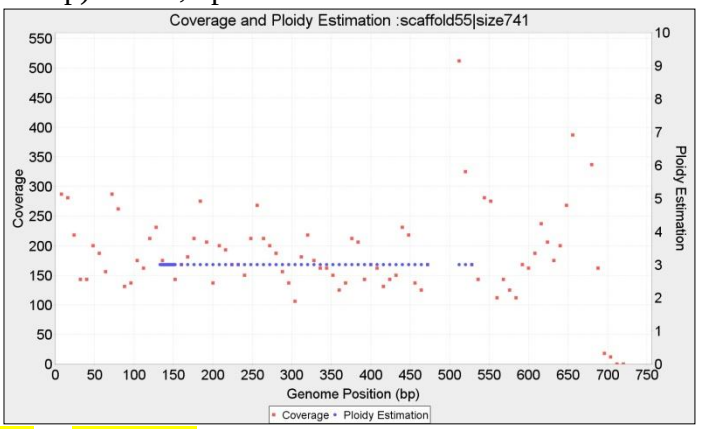
Complete ploidy estimation for Novogene CBS1483





Contig 56 (710 bp) *wl*=16, with p=1 and Contig 57 (657 bp) *wl*=16, p=1





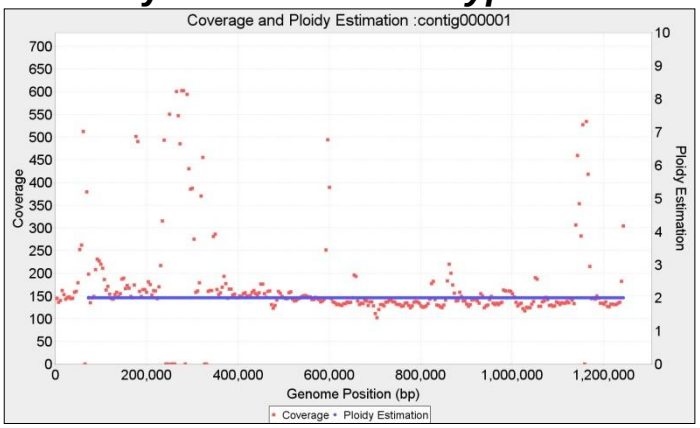
Contig 59 (526 bp) *wl*=16, p=2 and Contig 55 (741 bp) *wl*=20, p=3

Mapping of CBS1483 Scaffold, to *S.cerevisiae* S288 and *S.eubayanus* FM1318.

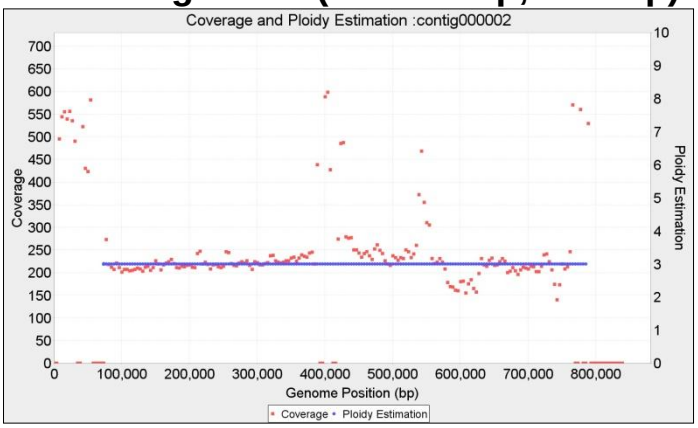
In order to evaluate our results we map the 59 contigs of the scaffold produced by (van den Broek et al., 2015) to S. *pastorianus* ancestors genomes. Sometimes, each scaffold maps to more than one ancestor chromosomes. We look at mappings in which at least 50% of the query scaffold aligns to at least 10% of the target ancestor chromosome. The following table shows the results. Headers: Percentage of the query scaffold mapped ('%Query') the name of the mapped scaffold ('VdB 2015 scaffold #') its size ('Q size'). Information from the target ancestor genome: percentage mapped (%Target) name of the reference chromosome and size. The last 7 columns display the ploidy estimations from the different methods. When a chromosome is fragmented with different ploidies, the estimations are order by decreasing length with p1 being the longest: 'VdB p1-p3' are the copy numbers estimated by Van den Broek et al. ; 'PEDCA Bc', is the result of PEDCA applied to the Baseclear sequencing. 'PEDCA Ng!=' signals only the estimation divergences between Baseclear and Novogene, both using PEDCA ('!=' different main ploidy from Baseclear, '+'additional fragment to Baseclear)

%Query mapped	VdB 2015 scaffold #	Q size (Kb)	%Target mapped	Target Reference	Reference size (Kbp)	VdB p1	VdB p2	VdB p3	PEDCA Bc_p1	PEDCA Bc_p2	PEDCA Bc_p3	PEDCA Ng !=
99	1	979.75	98.74	SeubIV	982.54	1	2		1			!=1
82	2	1038.13	80.5	SeubVII	1051.73	3			3	4		
18	2	1038.13	17.14	ScerVII	1090.94	1	4		3	4		
56	3	890.82	45.56	ScerVII	1090.94	1	4		1	5		
44	3	890.82	69.08	ScerVIII	562.64	5			1	5		
99	4	1460.99	94.57	ScerIV	1531.93	3	3		3			
100	5	777.50	95.49	ScerII	813.18	3			3			
98	6	745.80	98.53	SeubXV	741.89	1	2		1			
99	7	847.66	90.43	ScerXIII	924.43	2	4		2			
99	8	741.10	81.84	SeubXVI	896.11	2			2			
97	9	649.13	99.66	SeubXI	632.88	2			2			
98	10	1262.41	96.77	SeubII	1274.80	1	2	3	1			
99	11	637.73	94.77	ScerXI	666.82	2	3		2			
91	12	923.77	88.56	SeubXIII	953.69	2			2	4		
99	13	543.76	91.11	SeubV	588.91	2			2			
39	14	747.09	39.24	SeubX	747.93	1	2		2			
57	14	747.09	57.14	ScerX	745.75	2	1		2			
99	15	742.08	77.6	ScerXVI	948.07	2	4		2			
97	16	801.80	95.63	SeubVIII	813.83	1	2		1			
99	17	556.22	71.62	SeubXIV	768.02	2			2			
98	18	755.75	94.08	ScerXIV	784.33	2	3		2	3		
75	19	463.27	32.15	ScerXII	1078.18	2			2	3		
24	19	463.27	48.92	ScerI	230.22	3	4		2	3		
100	20	398.15	90.58	ScerIX	439.89	3			3			
98	21	823.16	73.82	ScerXV	1091.29	3			3	1		
99	22	452.43	43.17	SeubXII	1033.98	2	4		2			
99	23	443.17	40.6	ScerXII	1078.18	2			2			
64	24	652.26	55.43	SeubX	747.93	1	2		1			!=2
35	24	652.26	30.98	ScerX	745.75	2	1		1			!=2
98	25	347.59	31.31	ScerVII	1090.94	1	4		1			
100	26	392.54	97.34	SeubIX	401.36	2			2			
65	27	292.64	61.83	SeubIII	305.62	4						
35	27	292.64	32.17	ScerIII	316.62	4	3					
95	28	463.10	42.75	SeubXII	1033.98	2	4		2			
93	29	251.35	88.58	SeubVI	264.76	3			3	4		
95	30	570.60	93.97	ScerV	576.87	3			3			
94	31	208.41	17.89	ScerXV	1091.29	3			3			
99	32	149.73	84.69	SeubI	175.44	2	1		2			
56	33	147.91	7.64	ScerXII	1078.18	2			2	4	3	
40	33	147.91	25.78	ScerI	230.22	3	4		2	4	3	
89	34	259.47	85.75	ScerVI	270.16	1						
100	35	117.96	11.41	SeubXII	1033.98	2	4		2			+4
99	36	112.95	19.88	ScerVIII	562.64	5			5			
96	37	173.01	21.7	SeubXIV	768.02	2			2			
91	38	90.52	8.69	ScerXVI	948.07	2	4		4			
71	39	77.90	6.17	SeubXVI	896.11	2			4			
95	40	105.97	9.33	ScerXII	1078.18	2			2			
48	41	73.25	54.55	SeubMito	64.00				*			
99	42	28.80	3.73	SeubXIV	768.02	2			2			
56	43	28.77	2.74	SeubV	588.91	2			4			+5
91	44	19.65	10.17	SeubI	175.44	2	1		1			
63	45	12.55	1.35	SeubV	588.91	2			4			
100	47	2.41	0.22	ScerXII	1078.18	2			2			+1
100	49	1.69	0.16	SeubXII	1033.98	2	4		1			
77	50	1.64	0.08	ScerIV	1531.93	3			2	1		
93	51	0.99	0.06	ScerIV	1531.93	3			*			
91	52	0.95	0.06	ScerIV	1531.93	3						
100	53	0.94	0.21	ScerIX	439.89	3						
100	54	0.76	0.05	ScerIV	1531.93	3			1			
100	55	0.74	0.1	ScerX	745.75	2	1					
100	56	0.71	0.05	ScerIV	1531.93	3			1			
100	57	0.66	0.28	ScerI	230.22	3	4		1			
96	58	0.62	0.93	SeubMito	64.00							
95	59	0.59	0.95	ScerMito	85.00				2			

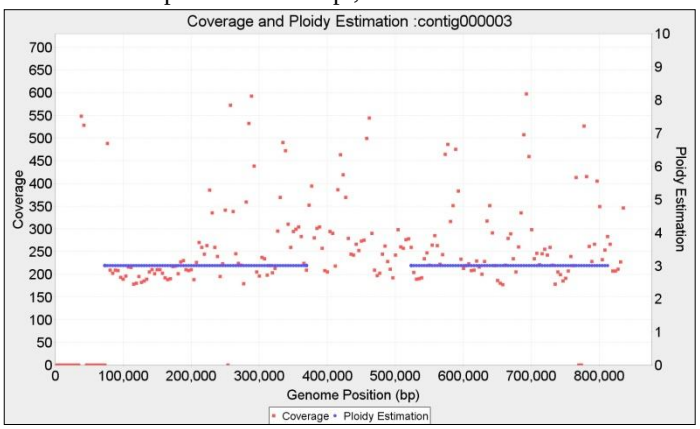
Ploidy estimation for *Trypanosoma cruzi* VI alignment (*wl*=1500bp, 2000bp)



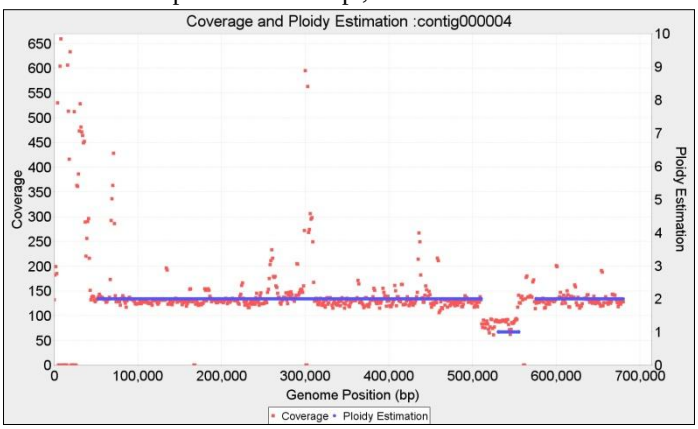
Chromosome1 p=2 *wl*=7.700 bp ;



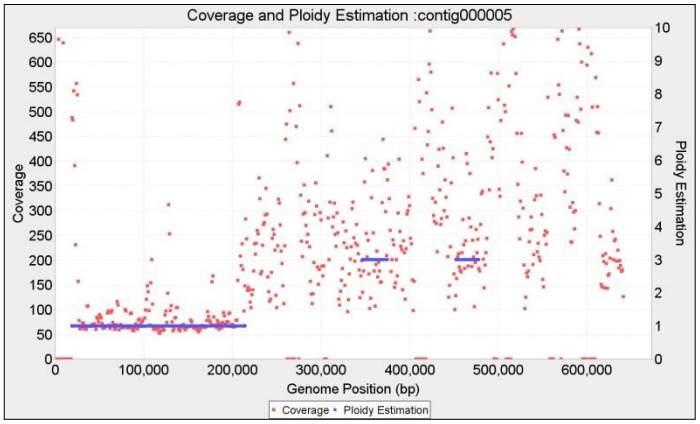
Chromosome2 p=3 *wl*=7.700 bp ;



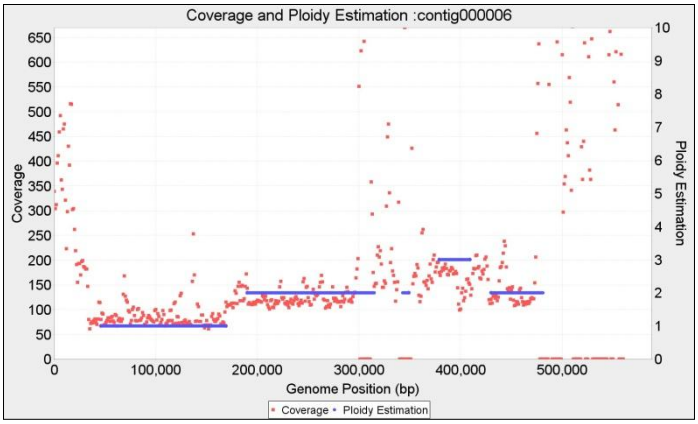
Chromosome3 p=3 *wl*=7.700 bp ;



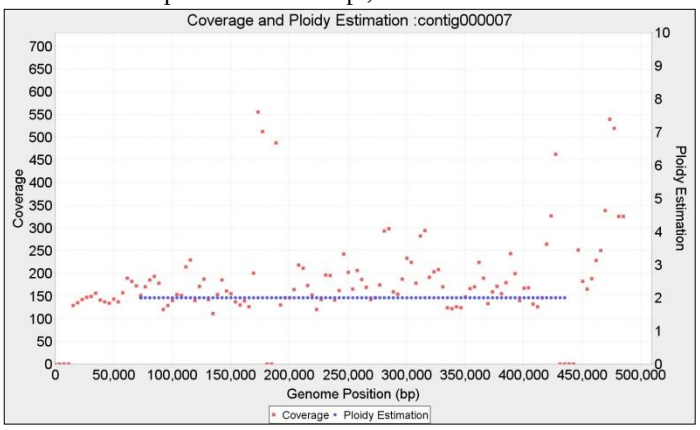
Chromosome4 p=2 *wl*=2.000 bp ;



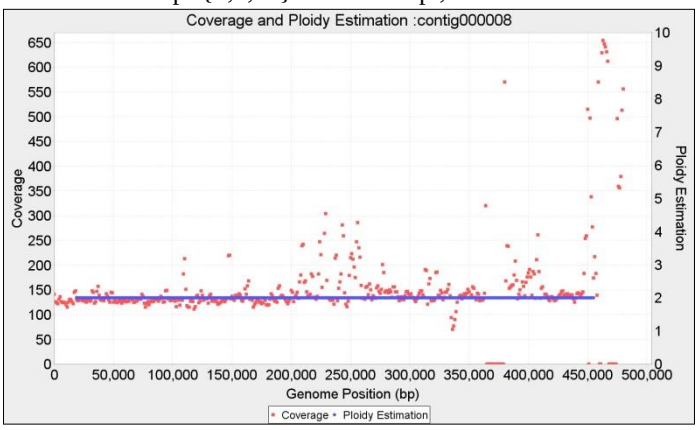
Chromosome5 p=2 *wl*=2.000 bp ;



Chromosome6 p={1,2,3?} *wl*=2.000 bp ;

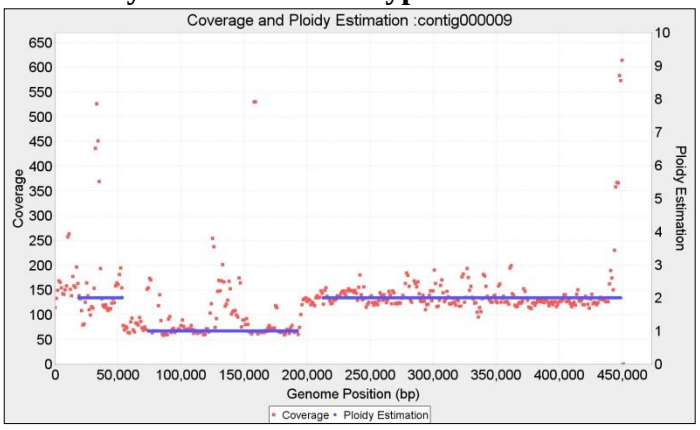


Chromosome7 p=2 *wl*=7.700 bp ;

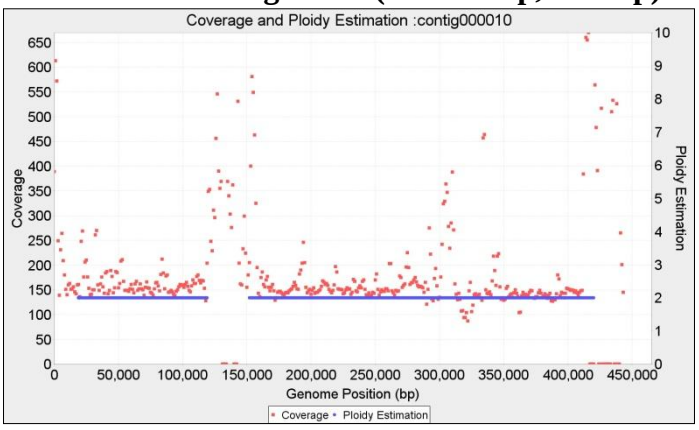


Chromosome8 p= *wl*=2.000 bp ;

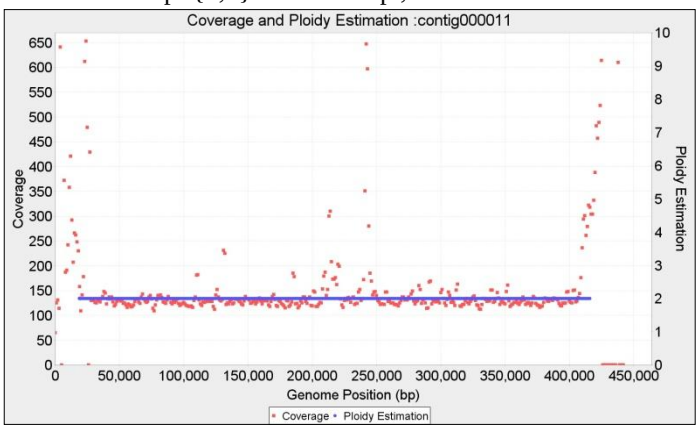
Ploidy estimation for Trypanosoma cruzi BroadTcVICLBrener alignment (wl=1500bp, 2000bp)



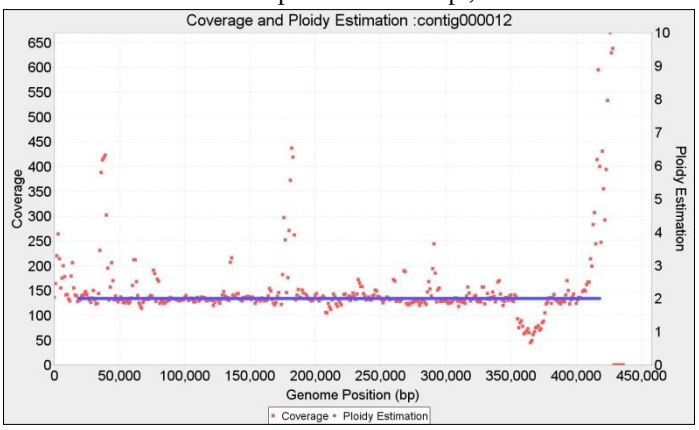
Chromosome9 p={2,1} wl=2.000 bp ;



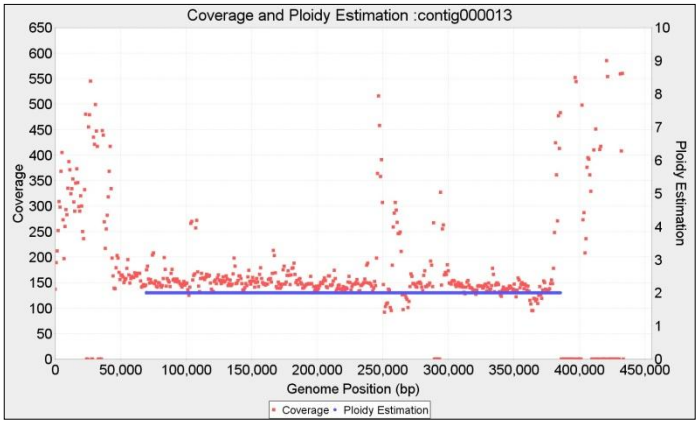
Chromosome10 p=2 wl=2.000 bp ;



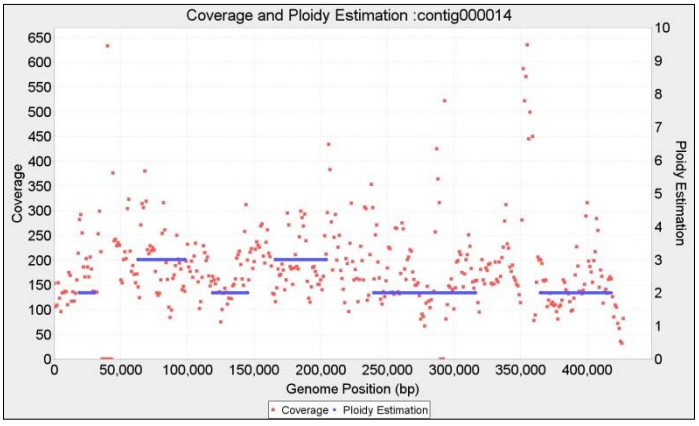
Chromosome11 p=2 wl=2.000 bp ;



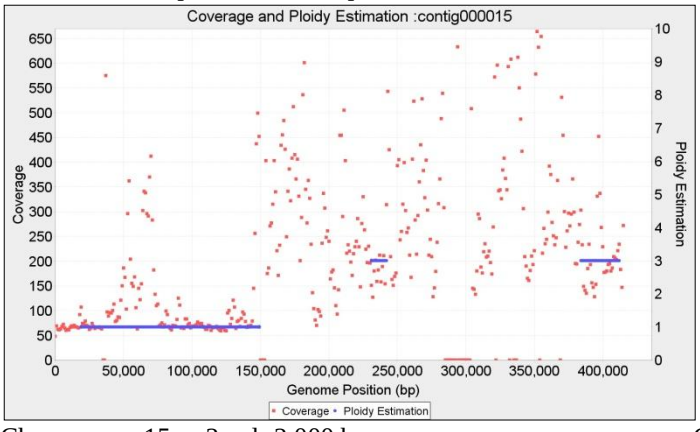
Chromosome12 p=2 wl=2.000 bp ;



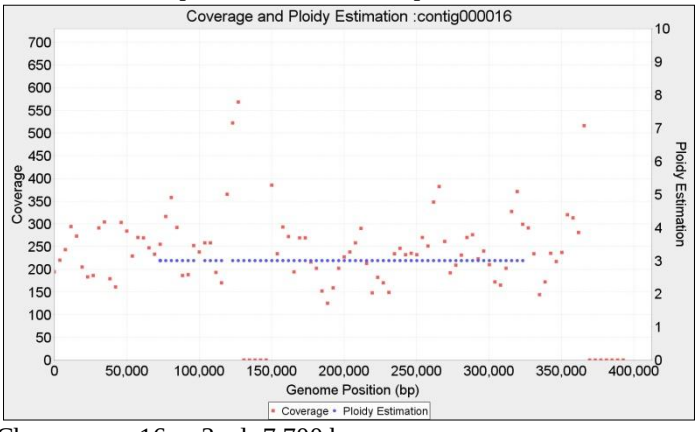
Chromosome13 p=2 wl=2.000 bp ;



Chromosome14 p={2,3?} wl=2.000 bp ;

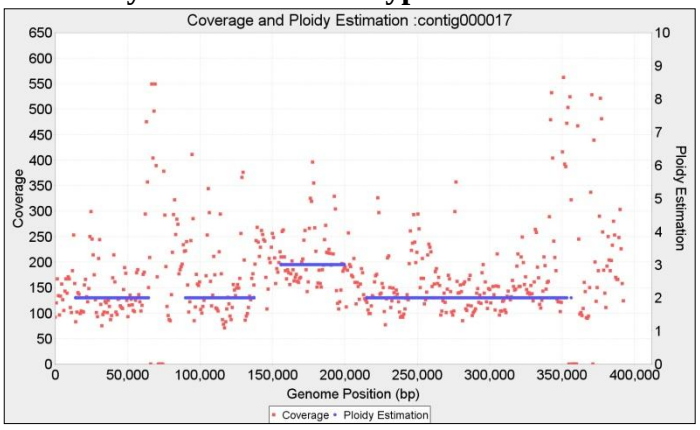


Chromosome15 p=2 wl=2.000 bp ;

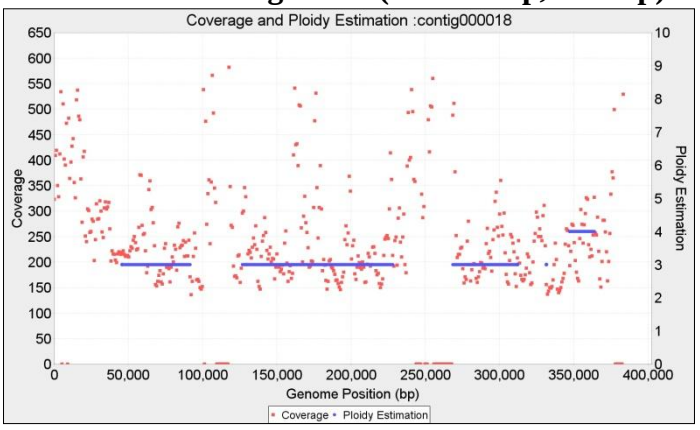


Chromosome16 p=3 wl=7.700 bp;

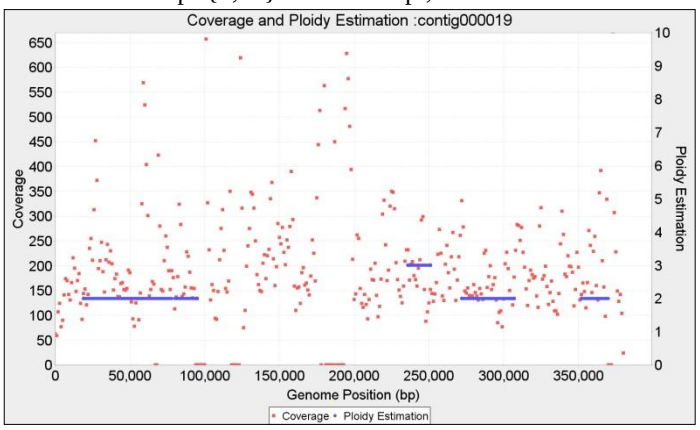
Ploidy estimation for Trypanosoma cruzi BroadTcVICLBrener alignment (wl=1500bp, 2000bp)



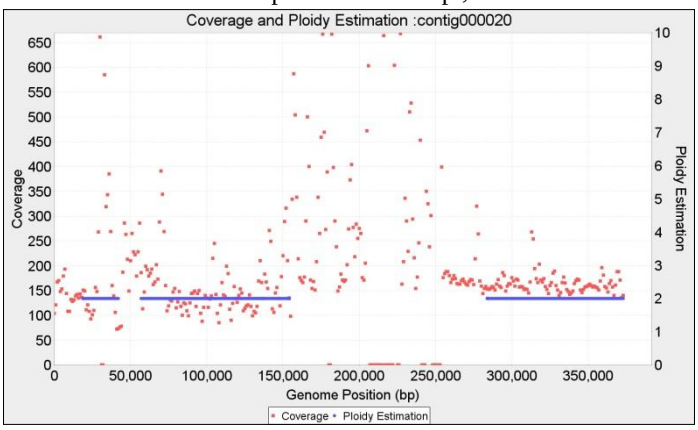
Chromosome17 p={2,3?} wl=2.000 bp ;



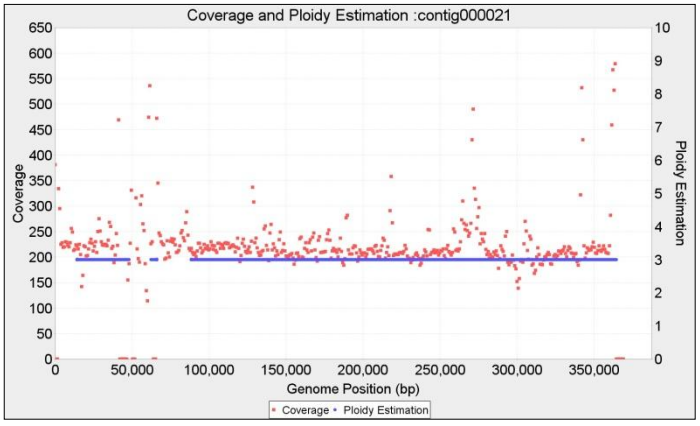
Chromosome18 p=3 wl=2.000 bp ;



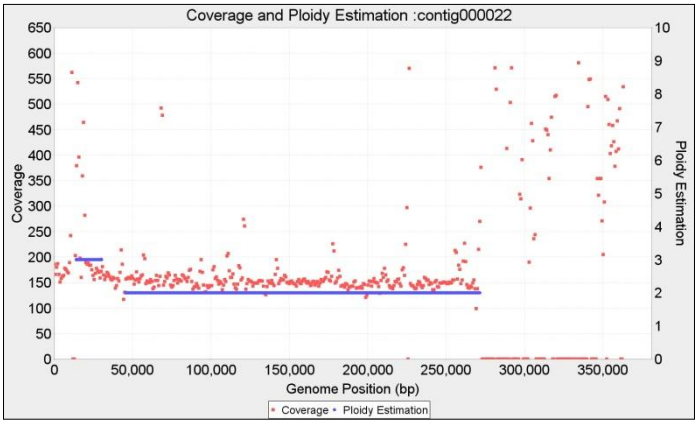
Chromosome19 p=2 wl=2.000 bp ;



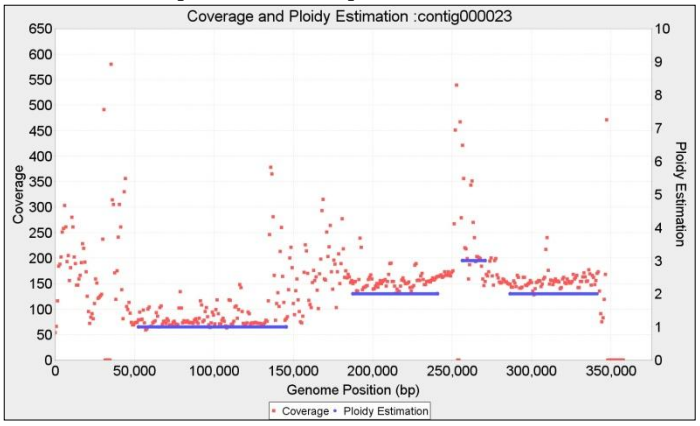
Chromosome20 p=2 wl=2.000 bp ;



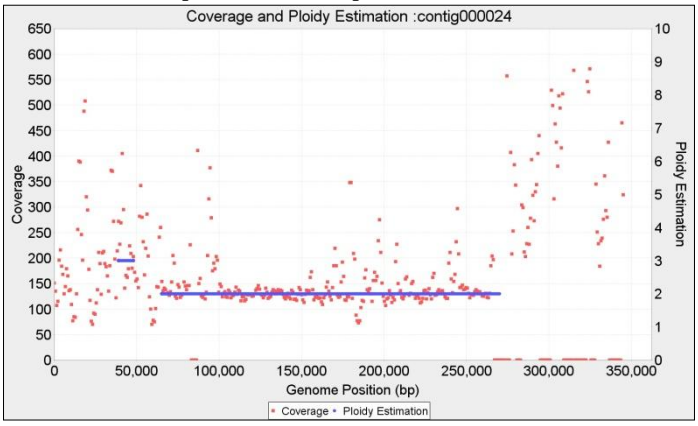
Chromosome21 p=3 wl=2.000 bp ;



Chromosome22 p=2 wl=2.000 bp ;

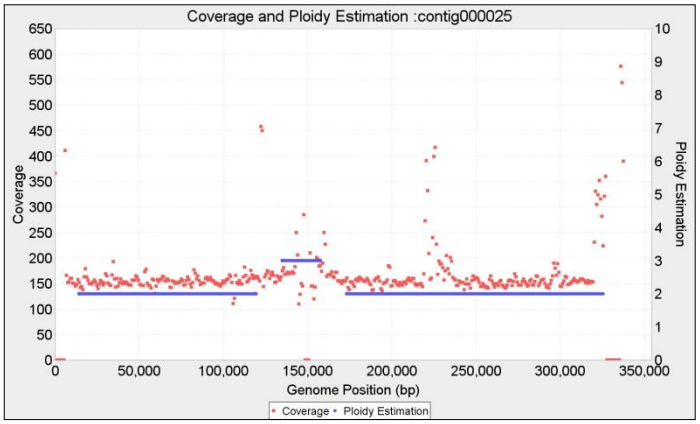


Chromosome23 p={1,2} wl=2.000 bp ;

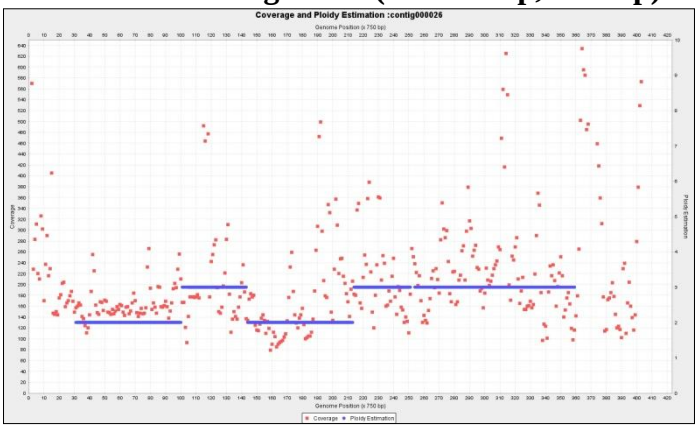


Chromosome24 p=2 wl=2.000 bp ;

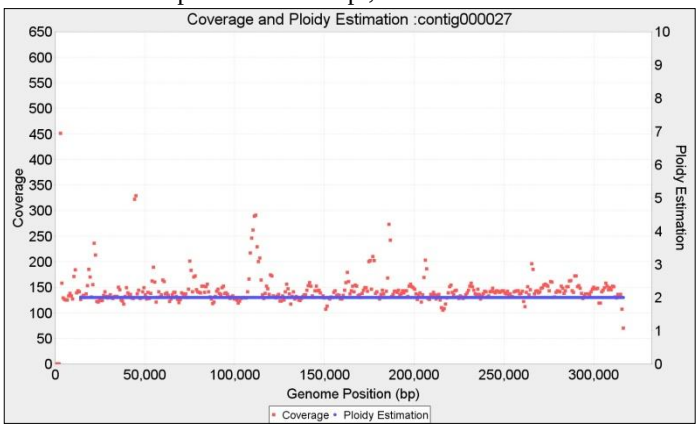
Ploidy estimation for Trypanosoma cruzi BroadTcVICLBrener alignment (wl=1500bp, 2000bp)



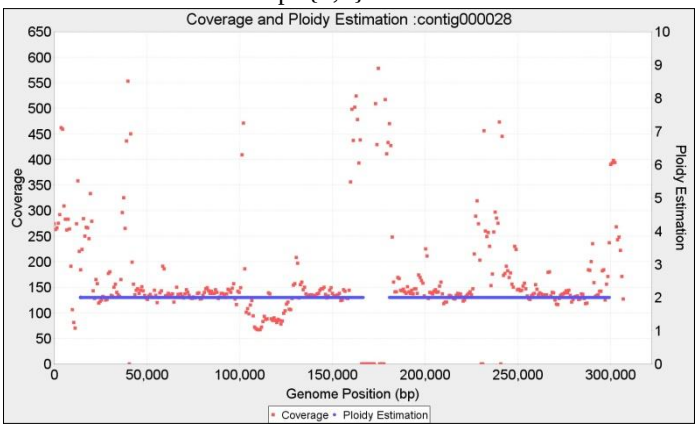
Chromosome25 p=2 wl=1.500 bp ;



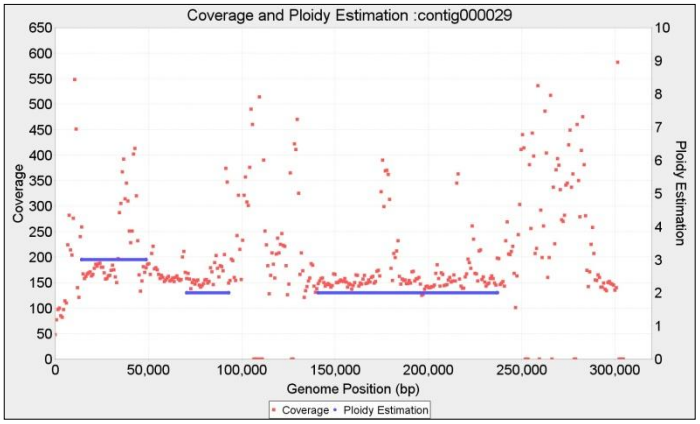
Chromosome26 p={2,3}



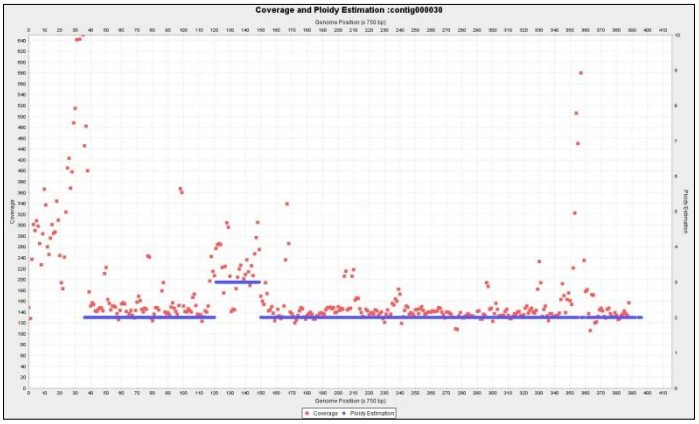
Chromosome27 p=2 wl=1.500 bp ;



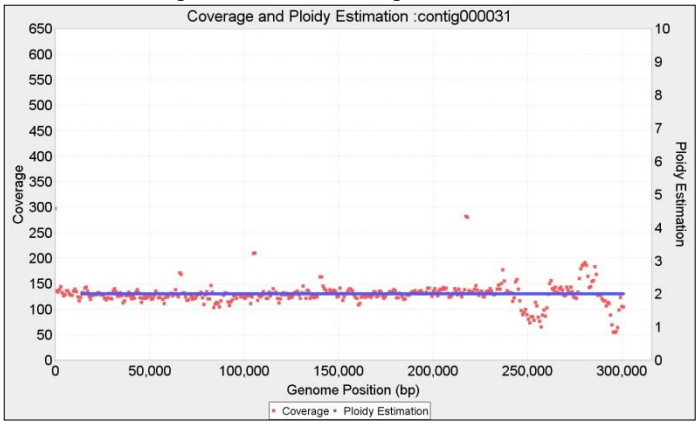
Chromosome28 p=2 wl=1.500 bp ;



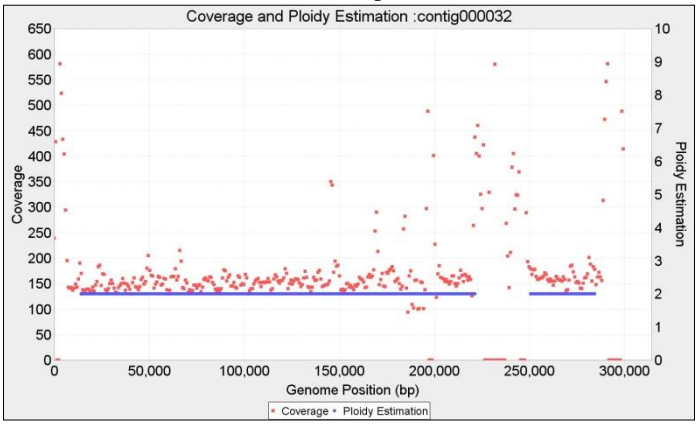
Chromosome29 p={3,2} wl=1.500 bp ;



Chromosome30 p={2,3} wl=

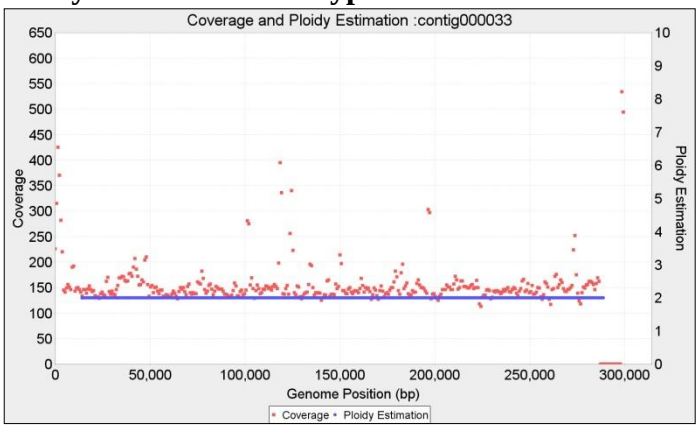


Chromosome31 p=2 wl=1.500 bp ;

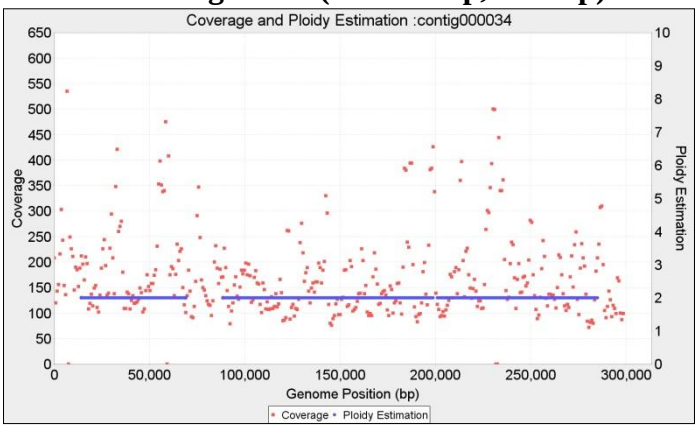


Chromosome32 p=2 wl=1.500 bp ;

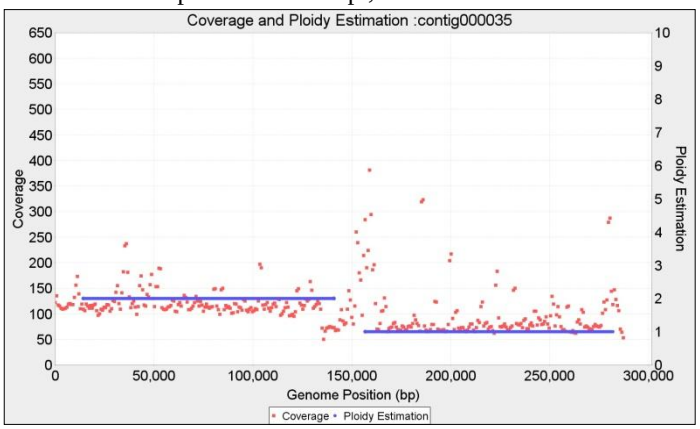
Ploidy estimation for Trypanosoma cruzi BroadTcVICLBrener alignment (wl=1500bp, 2000bp)



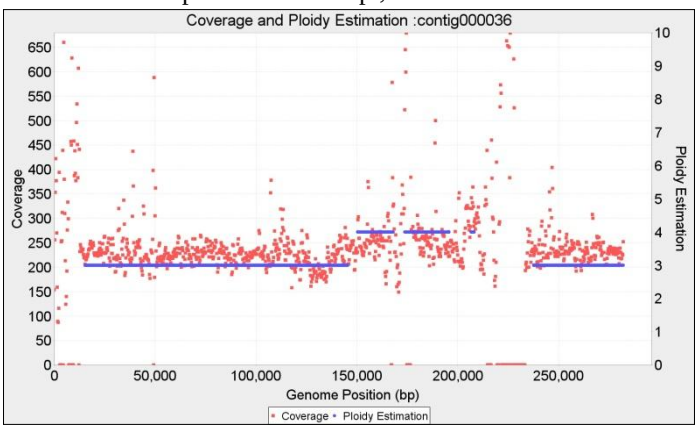
Chromosome33 p=2 wl=1.500 bp ;



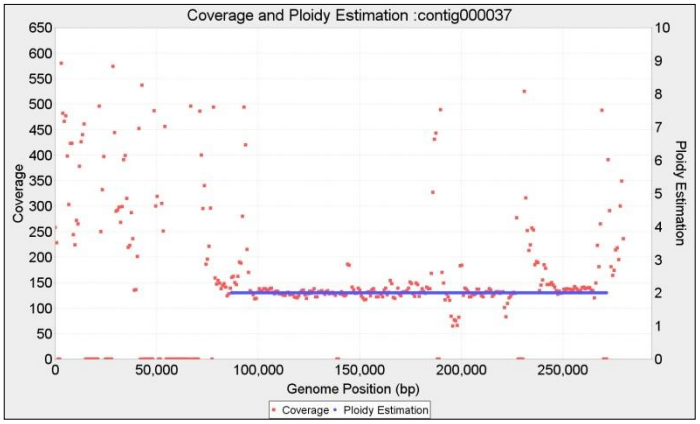
Chromosome34 p= 2 wl=1.500 bp ;



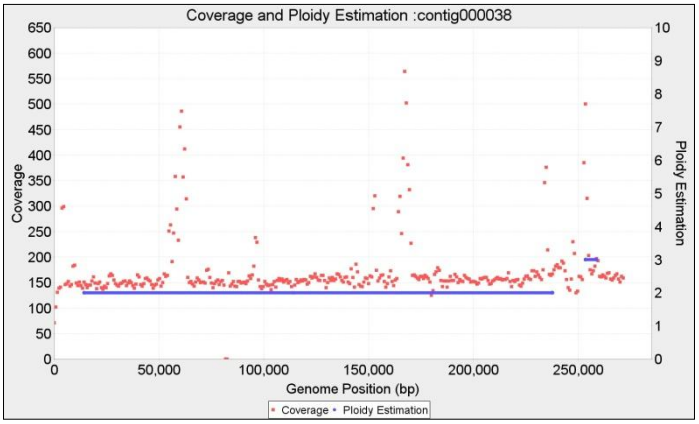
Chromosome35 p={2,1} wl=1.500 bp ;



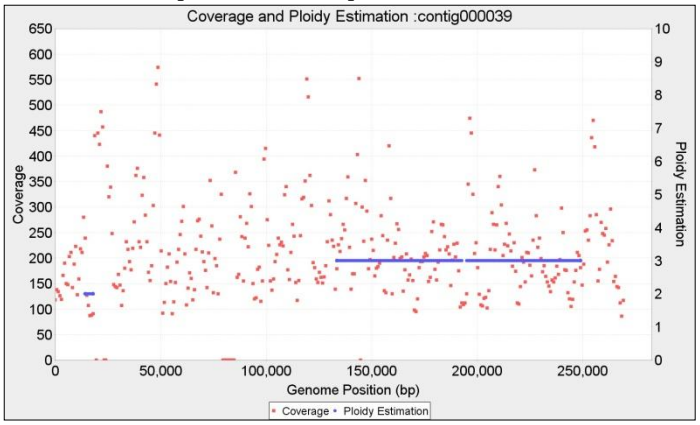
Chromosome36 p={3,4?} wl=500bp



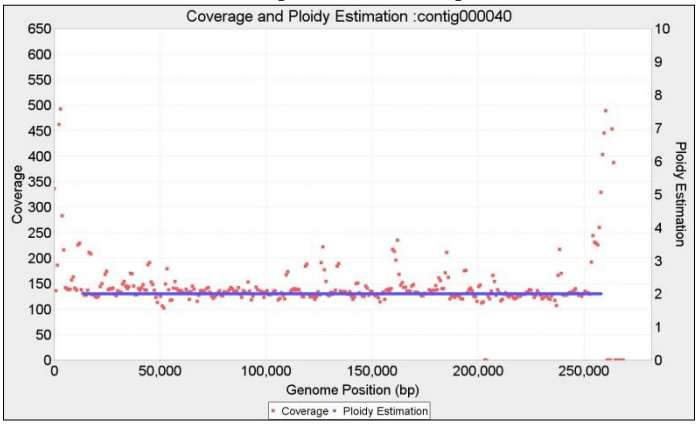
Chromosome37 p=2 wl=1.500 bp ;



Chromosome38 p=2 wl=1.500 bp ;

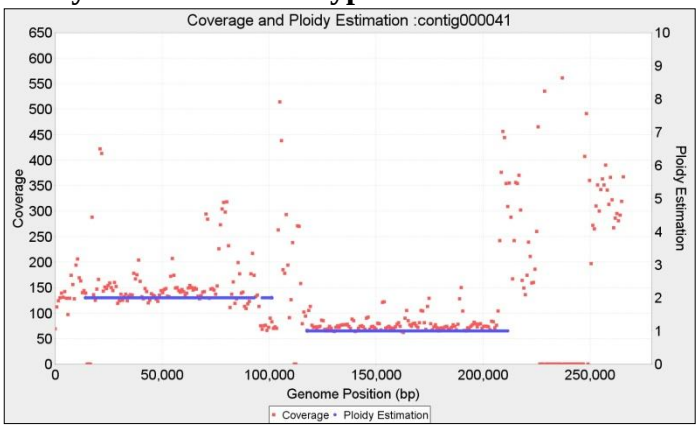


Chromosome39 p=3 wl=1.500 bp ;

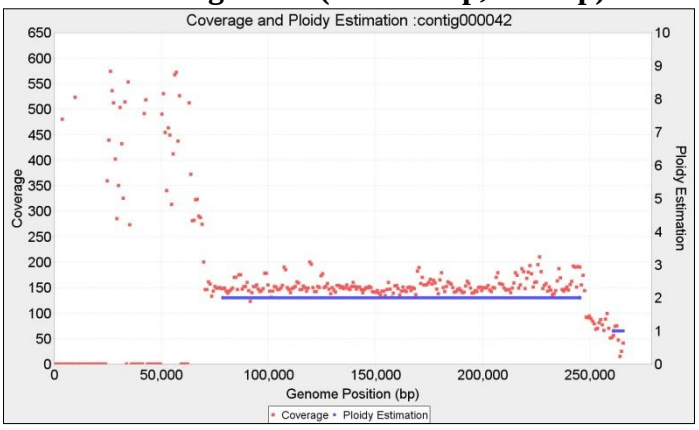


Chromosome40 p=2 wl=1.500 bp ;

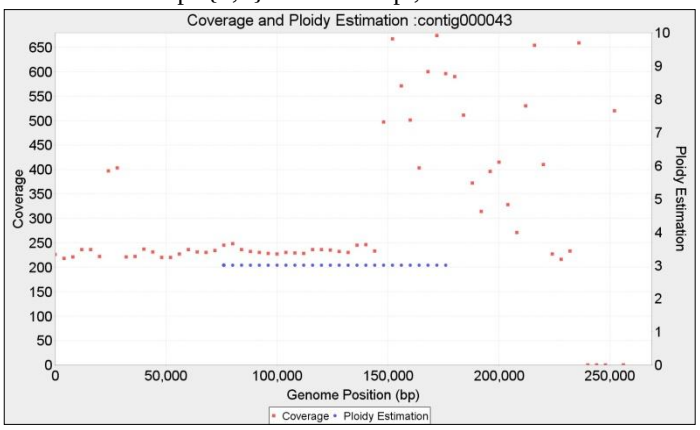
Ploidy estimation for Trypanosoma cruzi BroadTcVICLBrener alignment (wl=1500bp, 2000bp)



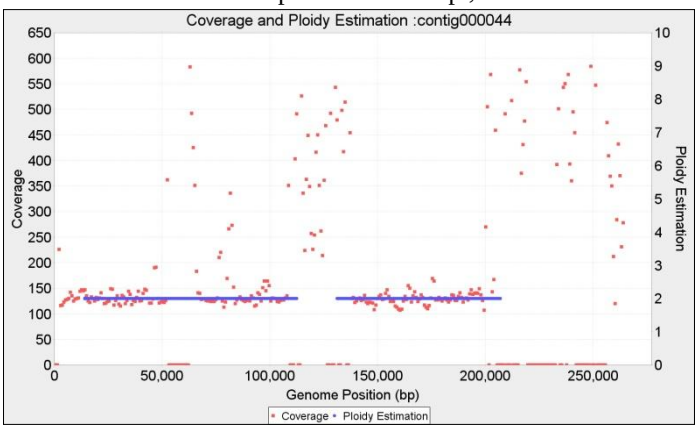
Chromosome41 p={2,1} wl=1.500 bp ;



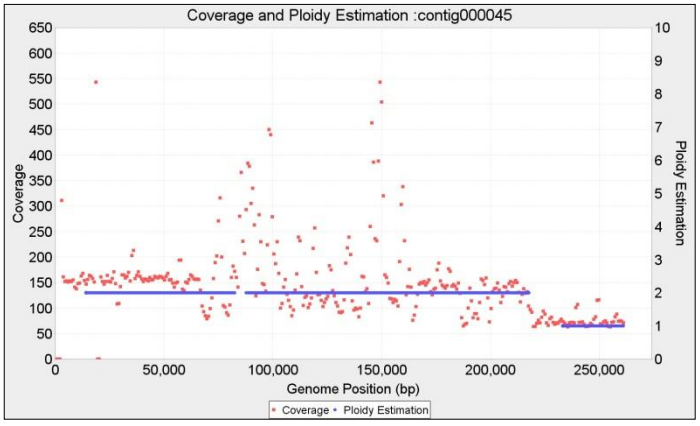
Chromosome42 p=2 wl=1.500 bp ;



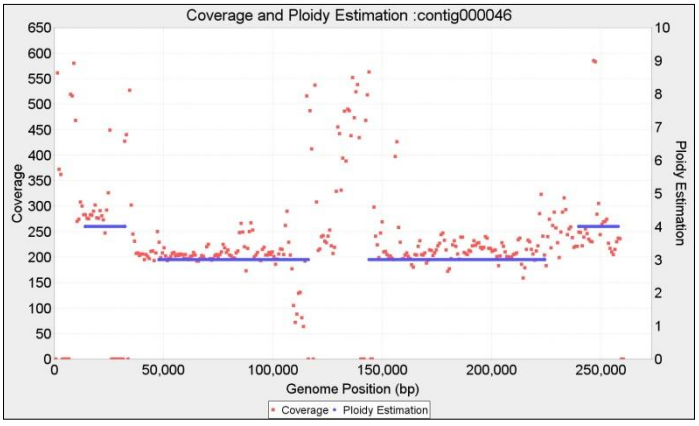
Chromosome43 p=3 wl=8.000 bp ;



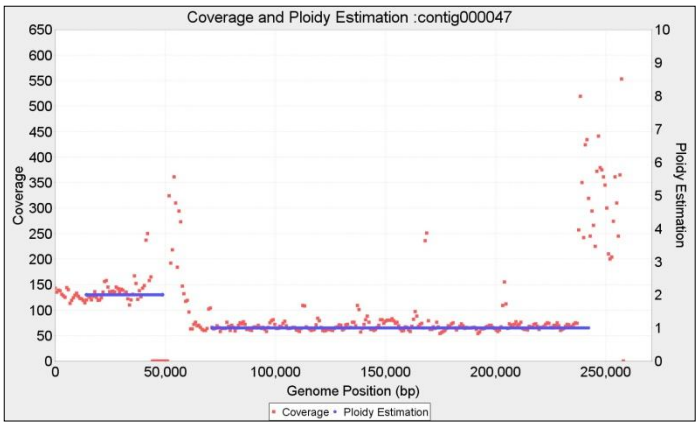
Chromosome44 p=2 wl=1.500 bp ;



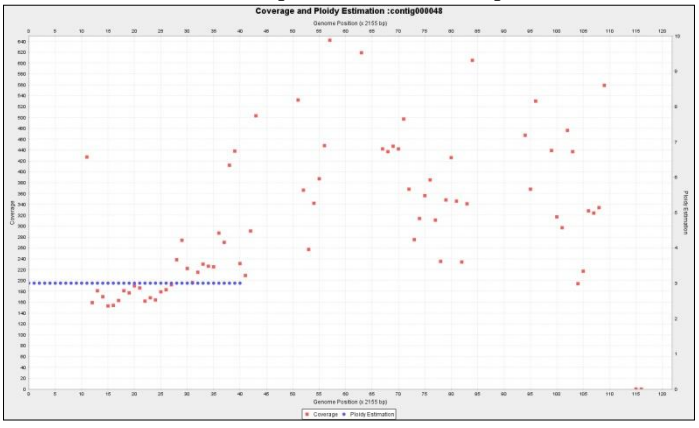
Chromosome45 p={2,1} wl=1.500 bp ;



Chromosome46 p={3,4} wl=1.500 bp ;

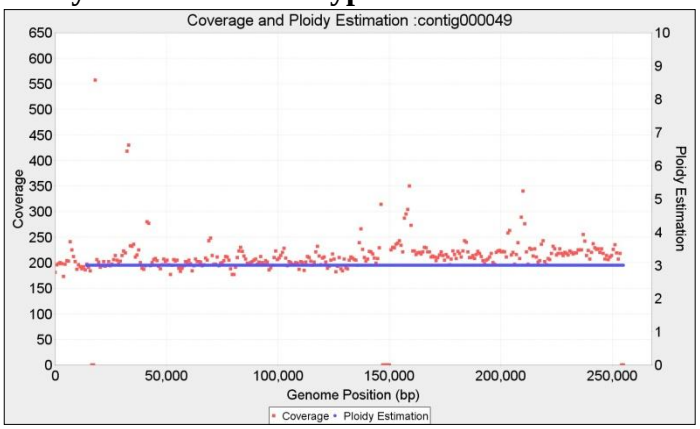


Chromosome47 p={2,1} wl=1.500 bp ;

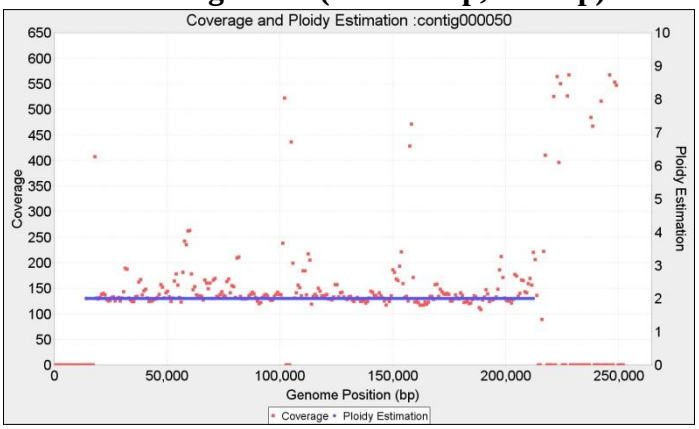


Chromosome48 p= 3 wl=1.500 bp ;

Ploidy estimation for Trypanosoma cruzi BroadTcVICLBrener alignment (wl=1500bp, 2000bp)



Chromosome49 p=3 wl=1.500 bp ;



Chromosome50 p=2 wl=1.500 bp ;