RESEARCH

The landscape of chloroplast genome assembly tools

Jan A Freudenthal[1,2†], Simon Pfaff[1,3†], Niklas Terhoeven[1,2], Arthur Korte[1], Markus J Ankenbrand[1,2,3^] and Frank Förster[1,3,4*^]

*Correspondence:
frank.foerster@uni-wuerzburg.de
[1]Center for Computational and Theoretical Biology, University of Würzburg, Campus Hubland Nord, 97074 Würzburg, Germany
Full list of author information is available at the end of the article
[†]Equal contributor ^Corresponding author

Abstract

Chloroplasts are photosynthetic organelles in plant cells and contain their own genomic information. That genome can be utilized in different scientific fields like phylogenetics or biotechnology. Thus, different assemblers have been developed specialized in chloroplast assemblies. Those assemblers often use the output of whole genome sequencing experiments as input. Such sequencing data usually contain the complete chloroplast genome information, even if the sequencing aims for the core genome. Different assembly tools have never been systematically compared. Here we present a benchmark of seven chloroplast assembly tools, capable to succeed in more than 60 % of real data sets. Our results show significant differences between the tested assemblers in terms of generating whole chloroplast genome sequences and computational requirements. Moreover, we suggest further development to improve user experience and success rate. In terms of reproducibility, we created docker images for each tested tool, which are available for the scientific community. Following the presented guidelines, users are able to analyze and screen data sets for chloroplast genomes using only standard computer infrastructure. Thus large scale screening for chloroplasts as hidden treasures within genomic sequencing data is feasible.

Keywords: Chloroplast; Genome; Assembly; Software; Benchmark

Introduction

General introduction and motivation

Chloroplasts are essential organelles present in plant cells and the cells of some protists. Chloroplasts enable the conversion of light energy into chemical energy via photosynthesis. They harbor their own ribosomes and a circular DNA genome usually with a size between 120 kbp to 160 kbp [1]. Because of this small size, the chloroplast genome has been an early target for sequencing. The first chloroplast genome sequences were obtained as early as 1986 [2, 3]. These early efforts elucidated the general genome organization and structure of the chloroplast DNA. Chloroplast genome content and structure are reviewed for example in [4, 5]. Chloroplast genomes are widely used for evolutionary analyses [6, 7], barcoding [8, 9, 10], and meta-barcoding [11, 12]. Interesting aspects of chloroplast genomes are their small size (120 kbp to 160 kbp,[1]), caused through endosymbiotic gene transfer [13, 14] and the low number of 100 to 120 genes that are still encoded on the chloroplast genome [4]. Despite the overall high conservation of the genome sequence, there are striking differences in the gene content between different groups (e.g. the loss of the whole *ndh* gene family in Droseraceae [15]). Even more extreme evolutionary cases,

where chloroplasts show a very low GC content and a modified genetic code are described [16].

These differences call for comparative genomic approaches. Given the small size, it is much easier to decipher the complete chloroplast genome than the complete core genome. For example the *Arabidopsis thaliana* core genome is approximately 125 Mbp in length [17, 18] while the size of the *A. thaliana* chloroplast genome with 154 kbp is more than 800 × smaller [19].

Even if only a single chloroplast is located inside a plant cell, several hundreds copies of the chloroplast genome exists in each cell [20, 21]. Therefore, many genome sequencing projects contain chloroplast reads as by-product. In some cases the chloroplast data is even considered contamination and experimental protocols for reducing their content have been developed [22]. An alternative approach to improve the assembly of the core genome would be to first resolve the chloroplast genome and afterwards use this information to remove those reads that map to the chloroplast genome.

Structurally, two inverted repeats (IR_A and IR_B) of 10 kbp to 76 kbp divide the chloroplast genome into a large (LSC) and a small single copy (SSC) region [1]. Those large inverted repeats complicate automated resolution with short read technologies[23]. Moreover, the existence of different chloroplasts within a single individual, and thus multiple different chloroplast genomes, have been described for different plants [24, 25, 26]. Although the origin and evolutionary importance of this phenomena —called heteroplasmy— are only poorly understood, the assembly of whole chloroplast genomes might be hindered.

Databases exist containing short read data for species where no reference chloroplast sequence is publicly available, eg. the Sequence Read Archive at NCBI [27]. The availability of whole chloroplast genomes would enable large scale comparative studies [28]. Additionally, reconstructed full chloroplast genomes have been used as super-barcodes [29], for biotechnology applications and genetic engineering [30].

Approaches to extracting chloroplasts from whole genome data

Different strategies have been developed to assemble chloroplast genomes [31]. In general, obtaining a chloroplast genome from WGS data requires two steps. First, the chloroplast reads have to be extracted from the mixed sequencing data. The second step is the assembly and resolution of the special circular structure including the inverted repeats. The extraction of the reads can be achieved by mapping the reads to a reference chloroplast. [32]. A different approach that does not perform alignments, relies on the higher coverage of chloroplast data in the whole genome sequencing data set[33]. Here, a k-mer analysis can be used to extract the most frequent reads. An example for this is implemented in `chloroExtractor` [34]. A third method combines both approaches by using a reference chloroplast as seed and simultaneously assembling the reads based on k-mers [35].

Purpose and scope of this study

The goal of this study is to compare the effectiveness and efficiency of existing open source command-line tools to de-novo assemble whole chloroplast genomes from raw genomic data sets with minimal configuration. This includes no need for

extensive data preparation, no need for a specific reference (apart from *A. thaliana*), no need to change default parameters, no manual finishing. We further restricted our benchmark to paired end Illumina data sets as these are routinely generated by modern sequencing platforms [36].

In our opinion this reflects the most common use cases: (1) a user trying a tool quickly without digging into options for fine tuning and (2) large scale automatic applications. Still, we acknowledge that the performance of the tools might be significantly improved by optimizing parameters (and references if applicable) for each data set specifically. However, an exhaustive comparison - including tuning of all different possible parameters for each tool- was out of scope for this study.

Our results will enable the discovery of novel chloroplast genomes as well as an assembly of inter/intra-individual differences in the respective chloroplast genomes.

Results

Performance metrics

Time requirements

In terms of run time, massive differences between the different tools have been observed. Apart from tool-specific differences, input data and number of threads had huge impact. The observed run times varied from a few minutes to several hours (figure 1).

Some assemblies failed to finish within the time-limit we set (48 h). On average the longest time to generate the assemblies was taken by `IOGA` and `Fast-Plast` followed by `ORG.Asm` and `GetOrganelle` the most time efficient tool was `chloroExtractor`, which on average is a little faster than `NOVOPlasty` and `Chloroplast assembly protocol`.

Not all the tools were able to benefit from having access to multiple threads. Both `NOVOPlasty` and `ORG.Asm` take about the same time independent of being able to utilize 1, 2, 4, or 8 threads. `Chloroplast assembly protocol`, `chloroExtractor`, `GetOrganelle`and `Fast-Plast` all profit from multi-threading (figures 1 and 2 and tables S3 to S5).

Memory and CPU Usage

The peak and mean CPU usage, as well as peak memory and disk usage have been recorded for all assemblers based on the same input data set and number of threads to use (figure 2 and tables S3 to S5). Mainly, the size of the input data influenced the peak memory usage with the exception of `chloroExtractor` and `IOGA`. Those two assemblers seems to have a memory usage pattern, which is less influenced by the size of the data. The number of allowed threads had only a limited impact on the peak memory usage. Nevertheless, all programs profit by a higher number of threads, if the size of the input data was increased. In contrast, the disk usage is independent from input size and number of threads for all assemblers.

Qualitative

The user experience of most tools was evaluated as mainly Good (table 1). However, a few critique points remained. Two minor dependencies were missing in the

`GetOrganelle` installation instructions and there was no test data available. Additionally, an issue occurred when running it on a *A. thaliana* data set. We are currently in the process of resolving this with the authors.

The `Fast-Plast` installation instructions were missing some dependencies. Like `GetOrganelle`, `Fast-Plast` does not offer a test data set or a tutorial, except for some example commands.

The `ORG.Asm` installation instructions did not work. We found some issues, which are probably related to the requirement of `Python 3.7`. There is a tutorial where sample data is available. However, following the instructions resulted in a segmentation fault. We found a workaround for this bug and contacted the authors.

The main critique point of `NOVOPlasty` was the lack of a test data set with instructions. This was fixed by the authors after we contacted them. Additionally, `NOVOPlasty` uses a custom license, where an OSI approved license would be preferred.

The `chloroExtractor` does come with a test data set and a short tutorial. However, it is currently not possible to evaluate the results of the test run.

The `IOGA` installation instructions were missing many dependencies. Also, there was no test data or tutorial available and there is no license assigned to it. Since there was no update to the GitHub repository for the last three years, the project can be seen as inactive. After contacting the authors, they promised to resolve the mentioned issues.

As many of the other tools, the installation instructions for the `Chloroplast assembly protocol` were missing some dependencies. The list was updated after we contacted the authors. This tool does come with a test data set, however a note about the expected outcome is missing. A more extensive tutorial is provided. The description about the parameter is short, but sufficient.

Quantitative

Simulated data

The only assembler obtaining perfect results according to our score for the simulated data sets is `GetOrganelle` (figure 3 and table 2). `IOGA` and `Chloroplast assembly protocol` showed the worst performance, being unable to fully assemble a single chloroplast out of 14 runs. `NOVOPlasty` performed second best with scores above 80 for all data sets, only failing to resolve the contigs into one single circular chloroplast assembly. The overall performance is best, when the input data consists purely of chloroplast reads. Only `IOGA` and `Chloroplast assembly protocol` failed to deliver any results under this scenario once. In general, no clear correlation between either length of the input reads or the ratio of core vs chloroplast reads and the performance of the different assemblers can be observed.

Real data sets

Concerning the performance of the assemblers on the real data sets, we were able to observe considerable differences in the median score (figure 4). The highest scores were achieved by `GetOrganelle` with a median of 99.7 and 199 circular assemblies out of a total of 356 assemblies that resulted in an output (table 3). The performance of `GetOrganelle` is followed by `Fast-Plast`, `NOVOPlasty`, `ORG.Asm`, and

`chloroExtractor`. `Fast-Plast` is outperforming the latter two slightly in terms of score, with twice as much 114 perfectly assembled chloroplast genomes (`NOVOPlasty` produced 66 and `ORG.Asm` 55 circular genomes). `IOGA` and `Chloroplast assembly protocol` were both not able to assemble a circular, single-contig genome (table 3), consequently resulting in the lowest mean and median scores (figure 5).

Consistency

Consistency was tested by re-running assemblies and comparison of the scores of two assemblies (figure 6). Replicates that did not produce an output were manually scored as 0. `GetOrganelle` was the only tool that succeeded in obtaining similar scores for all assemblies, without producing and completely unsuccessful assemblies for this subset of data. Except for `Fast-Plast` all the other tools had at least one assembly that was unsuccessful in one run, but produced an output in the other. Notably `IOGA` appears to have a tendency to perform differently in independent runs. Here, more than 10 % of the assemblies failed in one run only.

Both `Fast-Plast` and `NOVOPlasty` tend to have minor changes in the assembly when the overall performance is comparably well, leading to the arrow-shaped scatter plots. `chloroExtractor` and `Chloroplast assembly protocol` appear to be the most robust assemblers, having only few deviations between the two runs.

Discussion

We aimed to generate an overall performance score for the different chloroplast assemblers, but depending on distinct downstream applications, the different criteria assessed in this work need to be weighted differently. For example, ease of installation and use might not be a big concern if the tool is installed once and integrated in an automated pipeline. On the other hand this factor alone might prevent other users from being able to use the tool in the first place. Similarly, computational requirements or run time might be less relevant, if the goal is to assemble a single chloroplast for further analysis, but it is essential if hundreds or thousands of samples should be processed in parallel for a large scale study. Eventually, both ease of use and run time are irrelevant if the tool is not able to successfully accomplish its task. Also the scope of this study needs to be considered when interpreting the guidelines below. In particular, we evaluated all tools under the assumption that they are used in the most basic form (default parameters, no hand selected reference, no pre-processing of the data or post-processing of the result, restricted run time). It is important to note that any tool might perform significantly different, if the above mentioned parameters are fine-tuned for a specific data set.

The overall best success rate, both on simulated and real data, was achieved by `GetOrganelle` followed by `Fast-Plast`. Both tools complement each other, as each is able to successfully reconstruct a full chloroplasts in cases where the other tool fails. In rare cases `NOVOPlasty` or `ORG.Asm` are the only tool to succeed. The tools `Fast-Plast`, `NOVOPlasty`, and `ORG.Asm` produce the most variable results, thus re-running the tool after a failed attempt might be successful. `chloroExtractor` yields only few complete chloroplast assemblies, but requires also only few resources. It is easy to install and use and thus could be considered as a good option for a quick first try. Both `IOGA` and `Chloroplast assembly protocol` have the worst performance of all tools tested and fail to return reliable chloroplast assemblies.

Additionally, we observed no phylogenetic pattern in the success rate of the assemblers (figure 7). This indicates that the tools are generally able to reconstruct chloroplasts across the plant kingdom even without or with fixed *A. thaliana* as reference.

Guidelines for the end-user

Given these results, our recommendation is to use `GetOrganelle` as default option, and in case of failure `Fast-Plast` as backup solution. If both programs fail, it is sensible to re-run `Fast-Plast` and additionally try `NOVOPlasty` and `ORG.Asm`. This procedure maximizes the chance to effectively and efficiently recover the circular chloroplast genome from mixed genomic data. If none of these four assemblers produce sensible results, a reference guided approach and tweaking of the default parameters, might be the solution. Here, it is not possible to provide general guidelines, as the procedure will differ for different data sets. For an automated approach, running `GetOrganelle` and `Fast-Plast` in parallel appears to be a good trade-off between success rate and use of resources.

Ideas for future development

For further experiments, combining different components from different tools might be a promising approach. For example, read scaling from `chloroExtractor` followed by an assembly by `GetOrganelle` and finally the structural resolution with `Fast-Plast` could be a promising approach, combing the respective strength of the different tools.

Moreover, the installation issues need to be mitigated by modern software. Therefore, either containerization (docker, singularity, etc.) or install workflows (eg. bioconda [37]) should be established by all software packages. Otherwise, the burden of the software installation might result in scientists ignoring good tools.

Another important feature of software is a comprehensive documentation, which needs to be up-to-date and maintained. Additionally, software authors could improve the usability based on suggestions from their users.

Finally, all tools should improve their integrated guessing of default parameters, as many users avoid fine tuning of those, especially, for larger screening approaches. Last, as sequencing technology is developing fast (eg. PacBio or nanopore), tools need to be updated to not become obsolete. But the hope would be that with ongoing software development and improved sequencing technologies, the generation of whole chloroplast assemblies from any species will become a routine technique.

Conclusion

The main assumption for our study to benchmark different chloroplast assembly tools, is that whole genome sequencing data are also a promising source for chloroplast assemblies. Our benchmark shows that 60 % of the data sets without available chloroplast genome, have been assembled by at least one of the tools we analysed. Still, even with simulated (aka"perfect") data, not all tools succeeded in generating complete chloroplast assemblies. Therefore, we determined the strengths and weaknesses of the specific tools and provided guidelines for the users. However, it might be necessary, to combine different methods or manually explore the parameter space, to obtain reliable results if a single run seems not sufficient. Ultimately,

large scale studies reconstructing hundreds or thousands of chloroplast genomes are now feasible using the currently available tools.

Methods

Data availability

Source code for all methods used is available at [38] and archived in zenodo under [39]. All docker images are published on [40] and are named with a leading `benchmark_` (table 4).

To enable a fair comparison of all tools, we generated simulated sequencing data. Those simulated data sets are stored at [41]. This study adheres to the guidelines for computational method benchmarking [42].

Tool Selection

We included tools designed for assembling chloroplasts from whole genome paired end Illumina sequencing data. As a requirement, all tools must be available as open source software and allow execution via a command line interface. As a graphical user interface is not suitable for automated comparisons, tools only providing a graphical interface have not been included. The following tools were determined to be within the scope of this study: `ORG.Asm` [29], `chloroExtractor` [34], `Fast-Plast` [43], `IOGA` [44], `NOVOPlasty` [35], `GetOrganelle` [45], and `Chloroplast assembly protocol` [46].

Other related tools for assembling chloroplasts that did not meet our criteria and are therefore outside the scope of this study are for example: `Organelle PBA` [47], `sestaton/Chloro` [48], `Norgal` [49], and `MitoBim` [50].

`Organelle PBA` is designed for PacBio data and does not work with paired Illumina data alone. `sestaton/Chloro` fits our criteria, but it is flagged as work in progress and development and support seem to have ended two years ago. `Norgal` is a tool to extract organellar DNA from whole genome data based on a k-mer frequency approach. However the final output is a set of contigs of mixed mitochondrial and plastid origin. The suggested approach to get a finished chloroplast genome is to run `NOVOPlasty` on the ten longest contigs. Therefore we only included `NOVOPlasty` with the default settings and excluded `Norgal`. `MitoBim` is specifically designed for mitochondrial genomes. Even though there is a claim by the author that it can be used for chloroplasts as well, there is no further description on how to do that [51].

Additionally, there is a protocol for the `Geneious` [52] software available [53]. However, `Geneious` is closed source and GUI based, which is not in the scope of this study. There is also another publication describing a method for assembling chloroplasts [54]. However, the link to the software is not active anymore.

Our Setup

We want to use a minimum of different parameter settings for all assembly programs to enable a fair comparison. Therefore, we decided to specify that all programs have to work based on two input files, representing a data set's forward (`forward.fq`) and reverse (`reverse.fq`) sequence file in FASTQ format. Depending on the assembler, output files with different names and locations are generated. Those different

files are copied and renamed to ensure that each assembly approach produces the same output file (`output.fa`). Additionally, we set an environment variable for all programs to control the number of allowed threads. All three requirements (defined input file names, defined output file name, thread number control via environment variable) are ensured by a simple wrapper script (`wrapper.sh`). Finally, for a maximum of reproducibility all programs have been bundled into individual docker images based on a central base image which provides all the required software. Those docker images were used for the recording of the consumption of computational resources. Those docker images have been used for the performance benchmarking on a four Intel CPU-E7 8867 v3 system offering 1 TB of RAM. Furthermore, all our docker images have been converted into singularity containers for the quantitative measurement on simulated and real data sets. Singularity container were built from docker images for usage on a HPC-environment using Singularity v.2.5.2 [55] All singularity containers were run on Intel® Xeon® Gold 6140 Processors using a Slurm workload manager version 17.11.8 [56]. Assemblies were run on 4 threads using 10 GiB RAM with a time limit of 48 h.

Data

Simulated

To avoid suffering from sequencing errors and biological variances, we simulated perfect reads based on the *A. thaliana* (TAIR10) chloroplast assembly [57]. We used a sliding window approach with `seqkit` [58]. The exact commands are documented in `03_representative_datasets.md` in [41]. For the final simulated data sets reads based on mixtures of the *A. thaliana* (TAIR10) core and chloroplast genome were generated with different ratios (0:1, 1:10, 1:100, and 1:1000). Additionally, we generated data with different read lengths (150 bp and 250 bp). All data simulated contain exactly 2 million read pairs.

Real

We selected real data deposited at SRA [27]. We searched all data that matched `(((((((“green plants”[orgn]) AND “wgs”[Strategy]) AND “illumina”[Platform]) AND “biomol dna”[Properties]) AND “paired”[Layout]) AND “random”[Selection])) AND “public”[Access]` [59]. For each species with a reference chloroplast in CpBase [60], we selected one data set of those. In total, this accumulated to 369 data sets (table S1) representing a broad spectrum of the green plants (figure 7).

Evaluation Criteria

Computational Resources

We recorded the mean and the peak CPU usage, the peak memory consumption, and the size of the assembly folder for each program. As input data, we used different data sets comprising 25 000, 250 000 and 2 500 000 read pairs sampled from our simulated reads. We used our docker image setup (table 4) to run all assembly programs three times for each parameter setting. The different settings combined different input data and different number of threads to use (1, 2, 4 and 8).

Some programs will use more CPU threads than specified, therefore, the number of CPUs available have been fixed using the CPU option while running the docker run

command. For each assembly setting, we recorded the peak memory consumption, the CPU usage (mean and peak CPU usage) and the size of the folder where the assembly was calculated. The values of CPU and memory usage have been obtained by docker. The disk usage was estimated using the GNU tool `du`. We used `GNU parallel` for queuing of the different settings [61].

Qualitative

The qualitative evaluation is mainly based on the reviewer guidelines for the Journal of Open Source Software (JOSS) [62]. To create a standard environment, all tools were tested in a fresh default installation of Ubuntu 18.04.2 running in a virtual machine (VirtualBox Version 5.2.18_Ubuntu r123745). We chose this setup instead of the docker container, because it resembles a typical user environment better than the minimal docker installation. The tools were installed according to their installation instructions and the provided tutorial or example usage was executed. During the evaluation, the following questions were asked:

- Is the tool easy to install?
- Is there a way to test the installation or a tutorial on how to use the tool?
- Is there a good documentation on the parameter settings?
- Is the tool maintained (issues answered, implementation of new features)?
- Is the tool Open Source?

These questions were answered with Good, Okay or Bad, depending on the quality of the result. For example, a Good installation utilizes an automated package or dependency management like `apt`, CRAN, docker, etc. An Okay installation procedure provides a custom script to install everything or at least lists all dependencies. A Bad installation procedure fails to list important dependencies or produces errors, that prevent a successful installation without exhaustive debugging.

After an initial evaluation, we contacted all authors via their GitHub or GitLab issue tracking to communicate potential flaws we found.

Quantitative

For each data set and assembler the generated chloroplast genome was compared to the respective reference genome using a pairwise alignment obtained with `minimap2` v2.16 [63]. Based on theses alignments a score is calculated as shown in equation (1) The assemblies were scored on a scale from 0 to 100, with 100 being the best and 0 the worst possible score. Four different metrics were Incorporated, each contributing $\frac{1}{4}$ to the total score: Completeness, correctness, repeat resolution and continuity. These metrics are similar in concept by those used in the Assemblathon 2 project: coverage, validity, multiplicity, and parsimony [64].

The completeness is estimated as the coverage of the assembled chloroplast genome versus the reference genome (cov_{ref}) It resembles how many bases of the query genome can be mapped to its respective reference genome. Secondly, we mapped the reference genome against the query. The coverage of the reference genome (cov_{qry}) is used as measurement for the correctness of the assembly. The repeat resolution is estimated from the size difference of the assembly and the reference genome ($min\left\{\frac{cov_{qry}}{cov_{ref}}, \frac{cov_{ref}}{cov_{qry}}\right\}$), leading to values between 0 and 1. The fourth metric used is the continuity, represented by the number of contigs. A perfect score

is achieved if one circular chromosome was assembled, while the score gets worse with the amount of contigs.

$$score = \frac{1}{4} \cdot \left(cov_{ref} + cov_{qry} + min \left\{ \frac{cov_{qry}}{cov_{ref}}, \frac{cov_{ref}}{cov_{qry}} \right\} + \frac{1}{n_{contigs}} \right) \cdot 100 \quad (1)$$

Consistency

To ensure consistency of the obtained results, we randomly chose 100 data sets, that in the previous runs resulted in outputs for most of the assemblers and run them again with the same parameters as before. The resulting assemblies were scored again as described and the scores of the first and the second run were compared to each other This information is important to assess the robustness of the different programs.

Competing interests

The authors disclose that SP, NT, FF, and MJA are developers of `chloroExtractor`, one of the tools benchmarked in this article. The authors exercised caution not to let this fact unfairly influence their judgment and recommendations. JF, NT, and MJA are affiliated with the for-profit organization AnaLife Data Science.

Author's contributions

MJA and FF conceived of the presented idea and supervised the findings of this manuscript. SP and FF created the docker images. NT performed the qualitative analysis for all assemblers. MJA prepared the simulated and real data sets. JAF assembled the real data sets. FF ran the performance assemblies on the simulated data sets. All authors developed the score model. JAF and MJA implemented the score model and prepared the figures. All authors discussed the results and contributed to the final manuscript.

Availability of data and materials

The supplemental material is available from Zenodo [65]. The simulated data set is available from Zenodo [41]. All program code is available via Zenodo [39] or from Github [38]. The input data sets can be generated using the raw reads from NCBI SRA (links for each data set in table S1). The resulting assemblies are avaiable from Zenodo [66].

Author details

[1]Center for Computational and Theoretical Biology, University of Würzburg, Campus Hubland Nord, 97074 Würzburg, Germany. [2]AnaLife Data Science, Friedrich-Bergius-Ring 15, 97076 Würzburg, Germany. [3]Department of Bioinformatics, University of Würzburg, Biozentrum, Am Hubland, 97074, Würzburg, Germany. [4]Fraunhofer IME-BR, Winchester Str. 2, 97076 Gießen, Germany.

References

1. Palmer, J.D.: Comparative organization of chloroplast genomes. Annual Review of Genetics **19**(1), 325–354 (1985). doi:10.1146/annurev.ge.19.120185.001545. PMID: 3936406. https://doi.org/10.1146/annurev.ge.19.120185.001545
2. Ohyama, K., Fukuzawa, H., Kohchi, T., Shirai, H., Sano, T., Sano, S., Umesono, K., Shiki, Y., Takeuchi, M., Chang, Z., Aota, S.-i., Inokuchi, H., Ozeki, H.: Chloroplast gene organization deduced from complete sequence of liverwort marchantia polymorpha chloroplast DNA **322**(6079), 572. doi:10.1038/322572a0. Accessed 2019-05-20
3. Shinozaki, K., Ohme, M., Tanaka, M., Wakasugi, T., Hayashida, N., Matsubayashi, T., Zaita, N., Chunwongse, J., Obokata, J., Yamaguchi-Shinozaki, K., Ohto, C., Torazawa, K., Meng, B.Y., Sugita, M., Deno, H., Kamogashira, T., Yamada, K., Kusuda, J., Takaiwa, F., Kato, A., Tohdoh, N., Shimada, H., Sugiura, M.: The complete nucleotide sequence of the tobacco chloroplast genome: its gene organization and expression **5**(9), 2043–2049. doi:10.1002/j.1460-2075.1986.tb04464.x. Accessed 2019-05-20
4. Wicke, S., Schneeweiss, G.M., dePamphilis, C.W., Müller, K.F., Quandt, D.: The evolution of the plastid chromosome in land plants: gene content, gene order, gene function **76**(3), 273–297. doi:10.1007/s11103-011-9762-4. Accessed 2019-05-16
5. Green, B.R.: Chloroplast genomes of photosynthetic eukaryotes **66**(1), 34–44. doi:10.1111/j.1365-313X.2011.04541.x. Accessed 2019-05-16
6. Martín, M., Sabater, B.: Plastid ndh genes in plant evolution **48**(8), 636–645. doi:10.1016/j.plaphy.2010.04.009. Accessed 2016-06-12
7. Xiao-Ming, Z., Junrui, W., Li, F., Sha, L., Hongbo, P., Lan, Q., Jing, L., Yan, S., Weihua, Q., Lifang, Z., Yunlian, C., Qingwen, Y.: Inferring the evolutionary mechanism of the chloroplast genome size by comparing whole-chloroplast genome sequences in seed plants **7**(1), 1555. doi:10.1038/s41598-017-01518-5. Accessed 2019-05-16
8. Kress, W.J., Wurdack, K.J., Zimmer, E.A., Weigt, L.A., Janzen, D.H.: Use of DNA barcodes to identify flowering plants **102**(23), 8369–8374. doi:10.1073/pnas.0503123102. Accessed 2017-08-29
9. Hollingsworth, P.M., Forrest, L.L., Spouge, J.L., Hajibabaei, M., Ratnasingham, S., van der Bank, M., Chase, M.W., Cowan, R.S., Erickson, D.L., Fazekas, A.J., Graham, S.W., James, K.E., Kim, K.-J., Kress, W.J.,

Schneider, H., van AlphenStahl, J., Barrett, S.C.H., van den Berg, C., Bogarin, D., Burgess, K.S., Cameron, K.M., Carine, M., Chacón, J., Clark, A., Clarkson, J.J., Conrad, F., Devey, D.S., Ford, C.S., Hedderson, T.A.J., Hollingsworth, M.L., Husband, B.C., Kelly, L.J., Kesanakurti, P.R., Kim, J.S., Kim, Y.-D., Lahaye, R., Lee, H.-L., Long, D.G., Madriñán, S., Maurin, O., Meusnier, I., Newmaster, S.G., Park, C.-W., Percy, D.M., Petersen, G., Richardson, J.E., Salazar, G.A., Savolainen, V., Seberg, O., Wilkinson, M.J., Yi, D.-K., Little, D.P.: A DNA barcode for land plants **106**(31), 12794–12797. doi:10.1073/pnas.0905845106. Accessed 2019-05-21

10. de Vere, N., Rich, T.C.G., Trinder, S.A., Long, C.: DNA barcoding for plants **1245**, 101–118. doi:10.1007/978-1-4939-1966-6_8
11. Bell, K.L., Burgess, K.S., Okamoto, K.C., Aranda, R., Brosi, B.J.: Review and future prospects for DNA barcoding methods in forensic palynology **21**, 110–116. doi:10.1016/j.fsigen.2015.12.010. Accessed 2017-09-24
12. Deiner, K., Bik, H.M., Mächler, E., Seymour, M., Lacoursière-Roussel, A., Altermatt, F., Creer, S., Bista, I., Lodge, D.M., Vere, N.d., Pfrender, M.E., Bernatchez, L.: Environmental DNA metabarcoding: Transforming how we survey animal and plant communities **26**(21), 5872–5895. doi:10.1111/mec.14350. Accessed 2019-05-21
13. Martin, W., Rujan, T., Richly, E., Hansen, A., Cornelsen, S., Lins, T., Leister, D., Stoebe, B., Hasegawa, M., Penny, D.: Evolutionary analysis of arabidopsis, cyanobacterial, and chloroplast genomes reveals plastid phylogeny and thousands of cyanobacterial genes in the nucleus **99**(19), 12246–12251. doi:10.1073/pnas.182432999. Accessed 2019-05-20
14. Timmis, J.N., Ayliffe, M.A., Huang, C.Y., Martin, W.: Endosymbiotic gene transfer: organelle genomes forge eukaryotic chromosomes **5**(2), 123–135. doi:10.1038/nrg1271
15. Nevill, P.G., Howell, K.A., Cross, A.T., Williams, A.V., Zhong, X., Tonti-Filippini, J., Boykin, L.M., Dixon, K.W., Small, I.: Plastome-wide rearrangements and gene losses in carnivorous droseraceae **11**(2), 472–485. doi:10.1093/gbe/evz005. Accessed 2019-05-16
16. Su, H.-J., Barkman, T.J., Hao, W., Jones, S.S., Naumann, J., Skippington, E., Wafula, E.K., Hu, J.-M., Palmer, J.D., dePamphilis, C.W.: Novel genetic code and record-setting AT-richness in the highly reduced plastid genome of the holoparasitic plant balanophora **116**(3), 934–943. doi:10.1073/pnas.1816822116. Accessed 2019-05-20
17. Schmuths, H., Meister, A., Horres, R., Bachmann, K.: Genome size variation among accessions of arabidopsis thaliana. Annals of Botany **93**(3), 317–321 (2004)
18. Cao, J., Schneeberger, K., Ossowski, S., Günther, T., Bender, S., Fitz, J., Koenig, D., Lanz, C., Stegle, O., Lippert, C., *et al.*: Whole-genome sequencing of multiple arabidopsis thaliana populations. Nature genetics **43**(10), 956 (2011)
19. Sato, S., Nakamura, Y., Kaneko, T., Asamizu, E., Tabata, S.: Complete structure of the chloroplast genome of arabidopsis thaliana. DNA research **6**(5), 283–290 (1999)
20. Kumar, R.A., Oldenburg, D.J., Bendich, A.J.: Changes in DNA damage, molecular integrity, and copy number for plastid DNA and mitochondrial DNA during maize development. Journal of Experimental Botany **65**(22), 6425–6439 (2014). doi:10.1093/jxb/eru359. http://oup.prod.sis.lan/jxb/article-pdf/65/22/6425/16935653/eru359.pdf
21. Bendich, A.J.: Why do chloroplasts and mitochondria contain so many copies of their genome? BioEssays **6**(6), 279–282 (1987). doi:10.1002/bies.950060608. https://onlinelibrary.wiley.com/doi/pdf/10.1002/bies.950060608
22. Lutz, K.A., Wang, W., Zdepski, A., Michael, T.P.: Isolation and analysis of high quality nuclear DNA with reduced organellar DNA for plant genome sequencing and resequencing **11**, 54. doi:10.1186/1472-6750-11-54. Accessed 2019-05-20
23. Wang, W., Schalamun, M., Morales-Suarez, A., Kainer, D., Schwessinger, B., Lanfear, R.: Assembly of chloroplast genomes with long- and short-read data: a comparison of approaches using eucalyptus pauciflora as a test case. BMC genomics **19**(1), 977–977 (2018). doi:10.1186/s12864-018-5348-8. 30594129[pmid]
24. Corriveau, J.L., Coleman, A.W.: Rapid screening method to detect potential biparental inheritance of plastid dna and results for over 200 angiosperm species. American Journal of Botany **75**(10), 1443–1458 (1988)
25. Chat, J., Decroocq, S., Decroocq, V., Petit, R.J.: A Case of Chloroplast Heteroplasmy in Kiwifruit (Actinidia deliciosa) That Is Not Transmitted During Sexual Reproduction. Journal of Heredity **93**(4), 293–300 (2002). doi:10.1093/jhered/93.4.293. http://oup.prod.sis.lan/jhered/article-pdf/93/4/293/6454216/293.pdf
26. Scarcelli, N., Mariac, C., Couvreur, T.L.P., Faye, A., Richard, D., Sabot, F., Berthouly-Salazar, C., Vigouroux, Y.: Intra-individual polymorphism in chloroplasts from ngs data: where does it come from and how to handle it? Molecular Ecology Resources **16**(2), 434–445 (2016). doi:10.1111/1755-0998.12462. https://onlinelibrary.wiley.com/doi/pdf/10.1111/1755-0998.12462
27. Leinonen, R., Sugawara, H., Shumway, M., on behalf of the International Nucleotide Sequence Database Collaboration: The Sequence Read Archive. Nucleic Acids Research **39**(suppl_1), 19–21 (2010). doi:10.1093/nar/gkq1019. http://oup.prod.sis.lan/nar/article-pdf/39/suppl_1/D19/7624335/gkq1019.pdf
28. Tonti-Filippini, J., Nevill, P.G., Dixon, K., Small, I.: What can we do with 1000 plastid genomes? **90**(4), 808–818. doi:10.1111/tpj.13491. Accessed 2018-01-26
29. Coissac, E., Hollingsworth, P.M., Lavergne, S., Taberlet, P.: From barcodes to genomes: extending the concept of DNA barcoding **25**(7), 1423–1428. doi:10.1111/mec.13549. Accessed 2019-05-16
30. Daniell, H., Lin, C.-S., Yu, M., Chang, W.-J.: Chloroplast genomes: diversity, evolution, and applications in genetic engineering **17**. doi:10.1186/s13059-016-1004-2. Accessed 2019-05-20
31. Twyford, A.D., Ness, R.W.: Strategies for complete plastid genome sequencing **17**(5), 858–868. doi:10.1111/1755-0998.12626. Accessed 2018-01-26
32. Vinga, S., Carvalho, A.M., Francisco, A.P., Russo, L.M., Almeida, J.S.: Pattern matching through chaos game representation: bridging numerical and discrete data structures for biological sequence analysis. Algorithms for molecular biology : AMB **7**(1), 10–10 (2012). doi:10.1186/1748-7188-7-10. 22551152[pmid]
33. Chan, C.X., Ragan, M.A.: Next-generation phylogenomics. Biology direct **8**, 3–3 (2013). doi:10.1186/1745-6150-8-3. 23339707[pmid]

34. J Ankenbrand, M., Pfaff, S., Terhoeven, N., Qureischi, M., Gündel, M., L. Weiß, C., Hackl, T., Förster, F.: chloroExtractor: extraction and assembly of the chloroplast genome from whole genome shotgun data. The Journal of Open Source Software **3**(21), 464 (2018). doi:10.21105/joss.00464. Accessed 2019-05-22TZ
35. Dierckxsens, N., Mardulyn, P., Smits, G.: NOVOPlasty: de novo assembly of organelle genomes from whole genome data **45**(4), 18–18. doi:10.1093/nar/gkw955. Accessed 2018-01-26
36. Goodwin, S., McPherson, J.D., McCombie, W.R.: Coming of age: ten years of next-generation sequencing technologies. Nature Reviews Genetics **17**(6), 333–351 (2016). doi:10.1038/nrg.2016.49
37. Grüning, B., , Dale, R., Sjödin, A., Chapman, B.A., Rowe, J., Tomkins-Tinch, C.H., Valieris, R., Köster, J.: Bioconda: sustainable and comprehensive software distribution for the life sciences. Nature Methods **15**(7), 475–476 (2018). doi:10.1038/s41592-018-0046-7
38. GitHub Repository for Benchmark Project. https://github.com/chloroExtractorTeam/benchmark
39. Förster, F., Ankenbrand, M.J.: chloroExtractorTeam/benchmark: Benchmark Container Setup V2.0.1. doi:10.5281/zenodo.2628061. https://doi.org/10.5281/zenodo.2628061
40. Docker Hub Group for Benchmark Project. https://cloud.docker.com/u/chloroextractorteam/
41. Ankenbrand, M.J., Förster, F.: Simulated *Arabidopsis Thaliana* Sequencing Datasets for Chloroplast Assembler Benchmarking. doi:10.5281/zenodo.2622875. https://doi.org/10.5281/zenodo.2622875
42. Weber, L.M., Saelens, W., Cannoodt, R., Soneson, C., Hapfelmeier, A., Gardner, P., Boulesteix, A.-L., Saeys, Y., Robinson, M.D.: Essential guidelines for computational method benchmarking. 1812.00661. Accessed 2019-03-14
43. McKain, M., Afinit: Mrmckain/Fast-Plast: Fast-Plast V.1.2.6. Zenodo (2017). doi:10.5281/zenodo.973887. https://zenodo.org/record/973887 Accessed 2019-05-22TZ
44. Bakker, F.T., Lei, D., Yu, J., Mohammadin, S., Wei, Z., van de Kerke, S., Gravendeel, B., Nieuwenhuis, M., Staats, M., Alquezar-Planas, D.E., Holmer, R.: Herbarium genomics: plastome sequence assembly from a range of herbarium specimens using an Iterative Organelle Genome Assembly pipeline. Biological Journal of the Linnean Society **117**(1), 33–43 (2016). doi:10.1111/bij.12642. Accessed 2019-05-22TZ
45. Jin, J.-J., Yu, W.-B., Yang, J.-B., Song, Y., Yi, T.-S., Li, D.-Z.: GetOrganelle: a simple and fast pipeline for de novo assembly of a complete circular chloroplast genome using genome skimming data. bioRxiv (2018). doi:10.1101/256479. Accessed 2019-05-22TZ
46. Sancho, R., Cantalapiedra, C.P., López-Alvarez, D., Gordon, S.P., Vogel, J.P., Catalán, P., Contreras-Moreira, B.: Comparative plastome genomics and phylogenomics of *Brachypodium* : flowering time signatures, introgression and recombination in recently diverged ecotypes. New Phytologist **218**(4), 1631–1644 (2018). doi:10.1111/nph.14926. Accessed 2019-05-22TZ
47. Soorni, A., Haak, D., Zaitlin, D., Bombarely, A.: Organelle_pba, a pipeline for assembling chloroplast and mitochondrial genomes from pacbio dna sequencing data. BMC Genomics **18**(1), 49 (2017). doi:10.1186/s12864-016-3412-9
48. Staton, E.: Automated Chloroplast Genome Assembly. Accessed: 2019-05-28. ttps://github.com/sestaton/Chloro
49. Al-Nakeeb, K., Petersen, T.N., Sicheritz-Pontén, T.: Norgal: extraction and de novo assembly of mitochondrial dna from whole-genome sequencing data. BMC Bioinformatics **18**(1), 510 (2017). doi:10.1186/s12859-017-1927-y
50. Hahn, C., Bachmann, L., Chevreux, B.: Reconstructing mitochondrial genomes directly from genomic next-generation sequencing reads—a baiting and iterative mapping approach. Nucleic Acids Research **41**(13), 129–129 (2013). doi:10.1093/nar/gkt371. http://oup.prod.sis.lan/nar/article-pdf/41/13/e129/25367803/gkt371.pdf
51. MITObim Issue 16. Accessed: 2019-05-27. https://github.com/chrishah/MITObim/issues/16
52. Geneious Prime. Accessed: 2019-05-28. https://www.geneious.com
53. Gibbs, M.D.: De Novo Assembly and Reconstruction of Complete Circular Chloroplast Genomes Using Geneious Prime. Accessed: 2019-05-28. https://assets.geneious.com/documentation/geneious/App+Note+-+De+Novo+Assembly+of+Chloroplasts.pdf
54. Izan, S., Esselink, D., Visser, R.G.F., Smulders, M.J.M., Borm, T.: De novo assembly of complete chloroplast genomes from non-model species based on a k-mer frequency-based selection of chloroplast reads from total dna sequences. Frontiers in Plant Science **8**, 1271 (2017). doi:10.3389/fpls.2017.01271
55. Kurtzer, G.M., Sochat, V., Bauer, M.W.: Singularity: Scientific containers for mobility of compute. PloS one **12**(5), 0177459 (2017)
56. Jette, M.A., Yoo, A.B., Grondona, M.: Slurm: Simple linux utility for resource management. In: In Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP) 2003, pp. 44–60. Springer, ??? (2002)
57. Berardini, T.Z., Reiser, L., Li, D., Mezheritsky, Y., Muller, R., Strait, E., Huala, E.: The *Arabidopsis* information resource: Making and mining the "gold standard" annotated reference plant genome. genesis **53**(8), 474–485 (2015). doi:10.1002/dvg.22877. https://onlinelibrary.wiley.com/doi/pdf/10.1002/dvg.22877
58. Shen, W., Le, S., Li, Y., Hu, F.: Seqkit: A cross-platform and ultrafast toolkit for fasta/q file manipulation. PLOS ONE **11**(10), 1–10 (2016). doi:10.1371/journal.pone.0163962
59. SRA Search Term. Accessed: 2019-03-28. https://www.ncbi.nlm.nih.gov/sra/?term=(((((((%22green+plants%22%5Borgn%5D)+AND+%22wgs%22%5BStrategy%5D)+AND+%22illumina%22%5BPlatform%5D)+AND+%22biomol+dna%22%5BProperties%5D)+AND+%22paired%22%5BLayout%5D)+AND+%22random%22%5BSelection%5D))+AND+%22public%22%5BAccess%5D
60. Rocaps Lab: CpBase. Accessed: 2019-04-01, Version: 8/20/2017. http://rocaplab.ocean.washington.edu/old_website/tools/cpbase
61. Tange, O.: Gnu parallel - the command-line power tool. ;login: The USENIX Magazine **36**(1), 42–47 (2011). doi:10.5281/zenodo.16303
62. JOSS Review Criteria. Accessed: 2019-05-15. https://joss.readthedocs.io/en/latest/review_criteria.html

63. Li, H.: Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics **34**(18), 3094–3100 (2018)
64. Bradnam, K.R., Fass, J.N., Alexandrov, A., Baranay, P., Bechner, M., Birol, I., Boisvert, S., Chapman, J.A., Chapuis, G., Chikhi, R., Chitsaz, H., Chou, W.-C., Corbeil, J., Del Fabbro, C., Docking, T.R., Durbin, R., Earl, D., Emrich, S., Fedotov, P., Fonseca, N.A., Ganapathy, G., Gibbs, R.A., Gnerre, S., Godzaridis, E., Goldstein, S., Haimel, M., Hall, G., Haussler, D., Hiatt, J.B., Ho, I.Y., Howard, J., Hunt, M., Jackman, S.D., Jaffe, D.B., Jarvis, E.D., Jiang, H., Kazakov, S., Kersey, P.J., Kitzman, J.O., Knight, J.R., Koren, S., Lam, T.-W., Lavenier, D., Laviolette, F., Li, Y., Li, Z., Liu, B., Liu, Y., Luo, R., MacCallum, I., MacManes, M.D., Maillet, N., Melnikov, S., Naquin, D., Ning, Z., Otto, T.D., Paten, B., Paulo, O.S., Phillippy, A.M., Pina-Martins, F., Place, M., Przybylski, D., Qin, X., Qu, C., Ribeiro, F.J., Richards, S., Rokhsar, D.S., Ruby, J.G., Scalabrin, S., Schatz, M.C., Schwartz, D.C., Sergushichev, A., Sharpe, T., Shaw, T.I., Shendure, J., Shi, Y., Simpson, J.T., Song, H., Tsarev, F., Vezzi, F., Vicedomini, R., Vieira, B.M., Wang, J., Worley, K.C., Yin, S., Yiu, S.-M., Yuan, J., Zhang, G., Zhang, H., Zhou, S., Korf, I.F.: Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. GigaScience **2**(1) (2013). doi:10.1186/2047-217X-2-10. http://oup.prod.sis.lan/gigascience/article-pdf/2/1/2047-217X-2-10/25511257/13742_2013_article_29.pdf
65. Ankenbrand, M.J., Förster, F.: Supplemental Data for the Manuscript "The Landscape of Chloroplast Genome Assembly Tools". doi:10.5281/zenodo.3241963. https://doi.org/10.5281/zenodo.3241963
66. Ankenbrand, M.J., Förster, F.: Assemblies Based on Real Data Sets for the Manuscript "The Landscape of Chloroplast Genome Assembly Tools". doi:10.5281/zenodo.3240535. https://doi.org/10.5281/zenodo.3240535
67. Lex, A., Gehlenborg, N., Strobelt, H., Vuillemot, R., Pfister, H.: Upset: visualization of intersecting sets. IEEE transactions on visualization and computer graphics **20**(12), 1983–1992 (2014)
68. Federhen, S.: The ncbi taxonomy database. Nucleic acids research **40**(D1), 136–143 (2011)
69. Letunic, I., Bork, P.: Interactive tree of life (iTOL) v4: recent updates and new developments. Nucleic Acids Research (2019). doi:10.1093/nar/gkz239

Figures

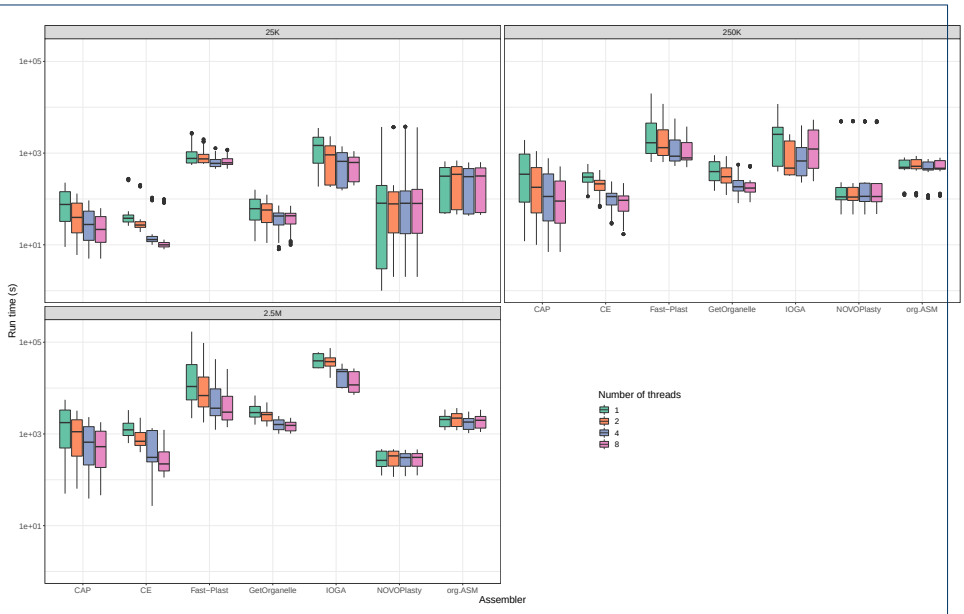

Figure 1 Computation time depending on number of threads and size of input data The boxplots show the differences in demand of CPU time for different number of threads and input data size for the seven different assemblers

Tables

Additional Files

Additional file 1 — supplemental data

Supplementary data contain a complete list of all real data sets used in this study. Additionally, a table with more details to the used docker images and the detailed results of the performance measurement are included. The file is available at [65].

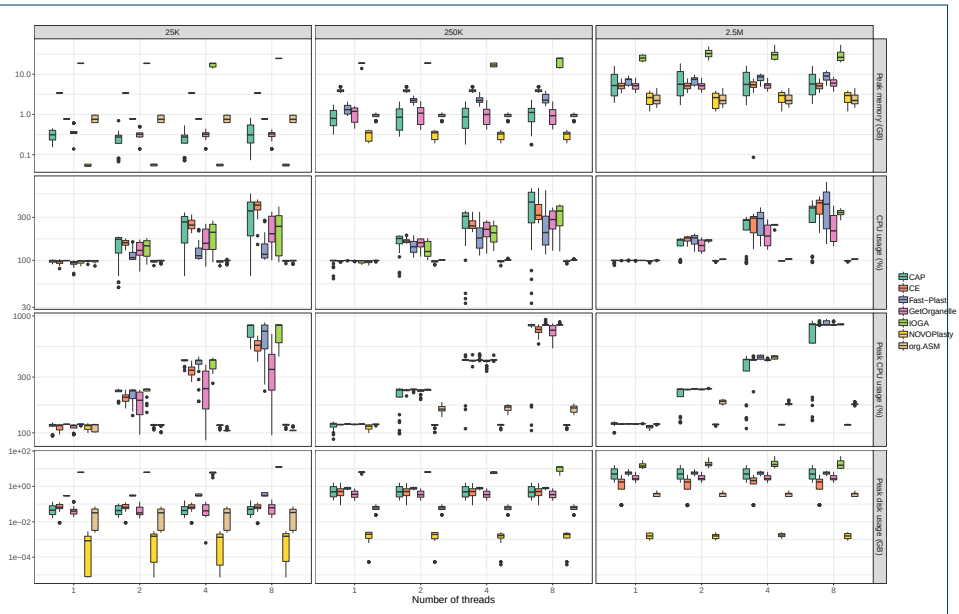

Figure 2 Performance metrics Boxplots depicting the demand of CPU and RAM and disk space needed depending on the assembler, input data size and number of threads

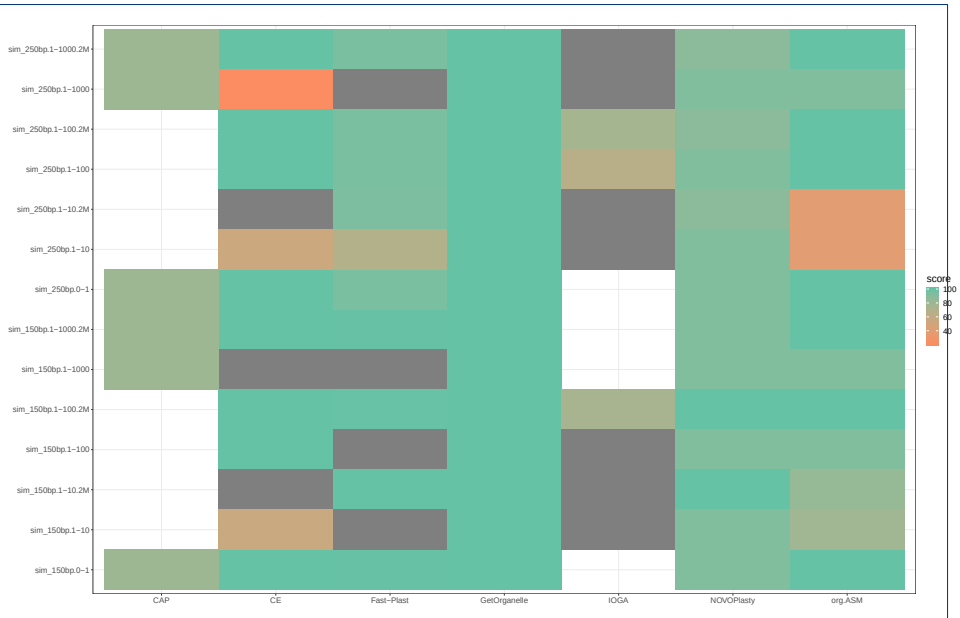

Figure 3 Score of assemblies on simulated data Results of assemblies from simulated data sets. Color scale of the tiles represents the score

Table 1 Overview of the results of the qualitative usability evaluation Each tool could score Good, Okay or Bad in each of the categories.

Tool	Installation	Test/Tutorial	Documentation	Maintenance	FLOSS
`chloroExtractor`	Good	Good	Good	Good	Good
`Chloroplast assembly protocol`	Okay	Good	Okay	Good	Good
`Fast-Plast`	Bad	Okay	Good	Good	Good
`GetOrganelle`	Okay	Okay	Good	Good	Good
`IOGA`	Bad	Bad	Okay	Okay	Bad
`NOVOPlasty`	Good	Good	Good	Good	Okay
`ORG.Asm`	Bad	Bad	Okay	Good	Good

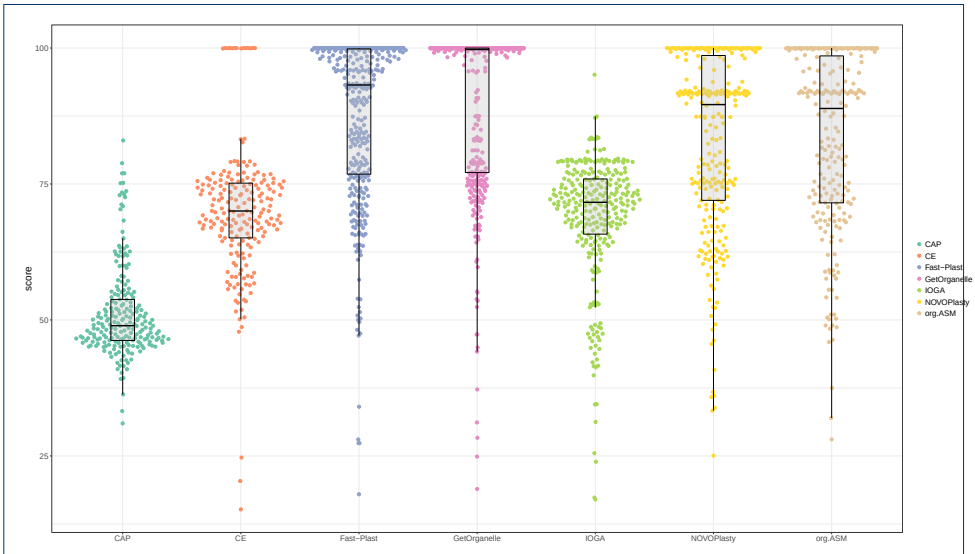

Figure 4 Results of scoring of the seven assemblers The box- and swarplots depict the results of the scoring algorithm we used. For the different assemblers. The whiskers of boxplots indicate the 1.5 x interquartile range.

Table 2 Scores of assemblies of simulated data

	data set	CAP	CE	Fast-Plast	GetOrganelle	IOGA	NOVOPlasty	org.ASM
1	sim_150bp.0-1	79.10	100.00	99.72	100.00		91.52	100.00
2	sim_150bp.1-10		56.44		100.00		91.52	78.00
3	sim_150bp.1-10.2M			99.97	100.00		100.00	82.72
4	sim_150bp.1-100		100.00		100.00		91.52	91.50
5	sim_150bp.1-100.2M		100.00	99.47	100.00	74.82	100.00	100.00
6	sim_150bp.1-1000	79.10			100.00		91.52	91.50
7	sim_150bp.1-1000.2M	79.10	100.00	99.72	100.00		91.52	100.00
8	sim_250bp.0-1	79.10	100.00	93.83	100.00		91.52	100.00
9	sim_250bp.1-10		54.98	68.45	100.00		91.52	40.20
10	sim_250bp.1-10.2M			93.00	100.00		87.40	40.20
11	sim_250bp.1-100		100.00	93.83	100.00	65.81	91.52	100.00
12	sim_250bp.1-100.2M		100.00	93.83	100.00	75.73	87.40	100.00
13	sim_250bp.1-1000	79.10	21.30		100.00		91.52	91.50
14	sim_250bp.1-1000.2M	79.10	100.00	93.83	100.00		87.40	100.00

Table 3 Mean scores of chloroplast genome assemblers

	assembler	Mean	SD	N_perfect	N_tot
1	CAP	51.26	8.42	0	221
2	CE	69.99	12.33	14	205
3	Fast-Plast	86.96	15.33	114	352
4	GetOrganelle	89.13	15.30	199	356
5	IOGA	68.74	11.69	0	296
6	NOVOPlasty	82.78	16.55	66	270
7	org.ASM	82.77	16.38	55	228

Table 4 Docker images used in our benchmark setup SHA256 checksums are stated in table S2

Tool	Image name and tag
`chloroExtractor`	chloroextracteam/benchmark_chloroextractor:v2.0.0
`Chloroplast assembly protocol`	chloroextracteam/benchmark_chloroplast_assembly_protocol:v2.0.1
`Fast-Plast`	chloroextracteam/benchmark_fastplast:v2.0.0
`GetOrganelle`	chloroextracteam/benchmark_getorganelle:v2.0.0
`IOGA`	chloroextracteam/benchmark_ioga:v2.0.0
`NOVOPlasty`	chloroextracteam/benchmark_novoplasty:v2.0.0
`ORG.Asm`	chloroextracteam/benchmark_org-asm:v2.0.0

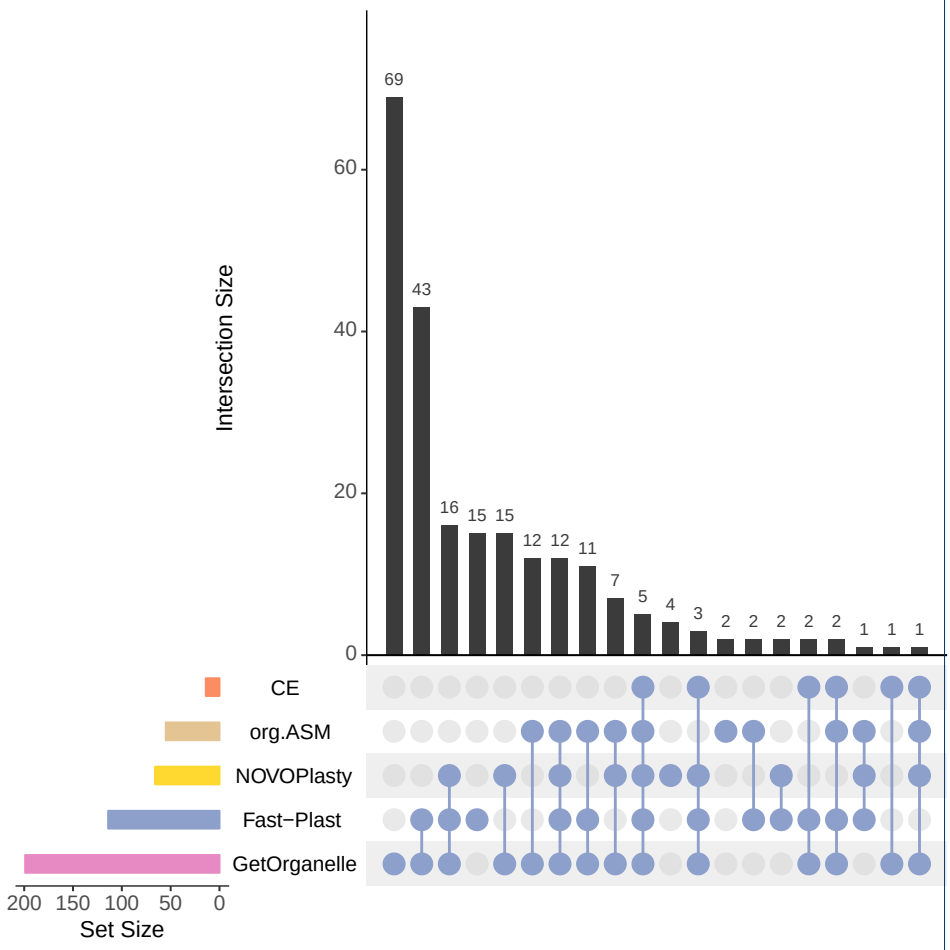

Figure 5 Upset plot [67] comparing success of assemblers on the real data sets The plot shows the intersection of success ($score > 99$) between assemblers. For 69 data sets only `GetOrganelle` was able to obtain a complete chloroplast. 43 were successful with both `GetOrganelle` and `Fast-Plast` and so on

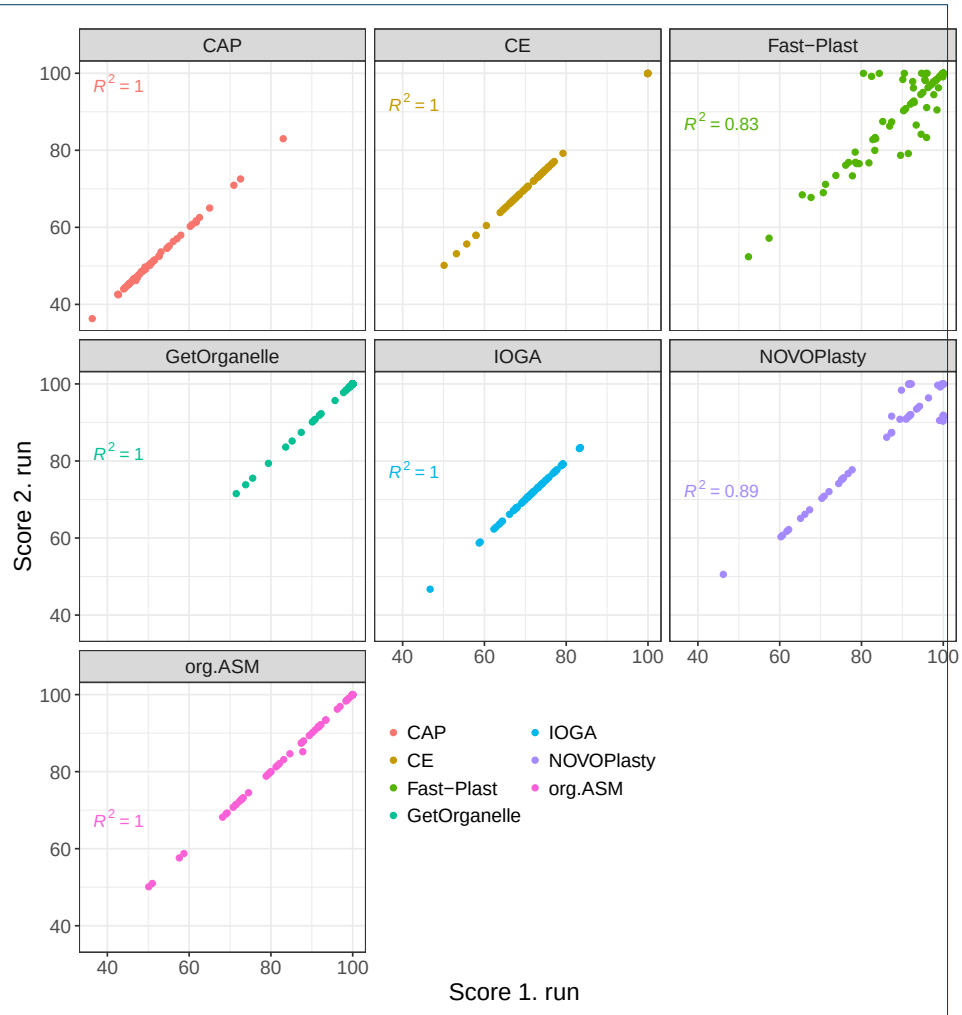

Figure 6 Scores between two repeated runs for consistency testing The scatter plots depicts the scores of the 1. runs x-axis versus the scores of the 2. run y-axis of the data sets that were selected for re-evaluation.

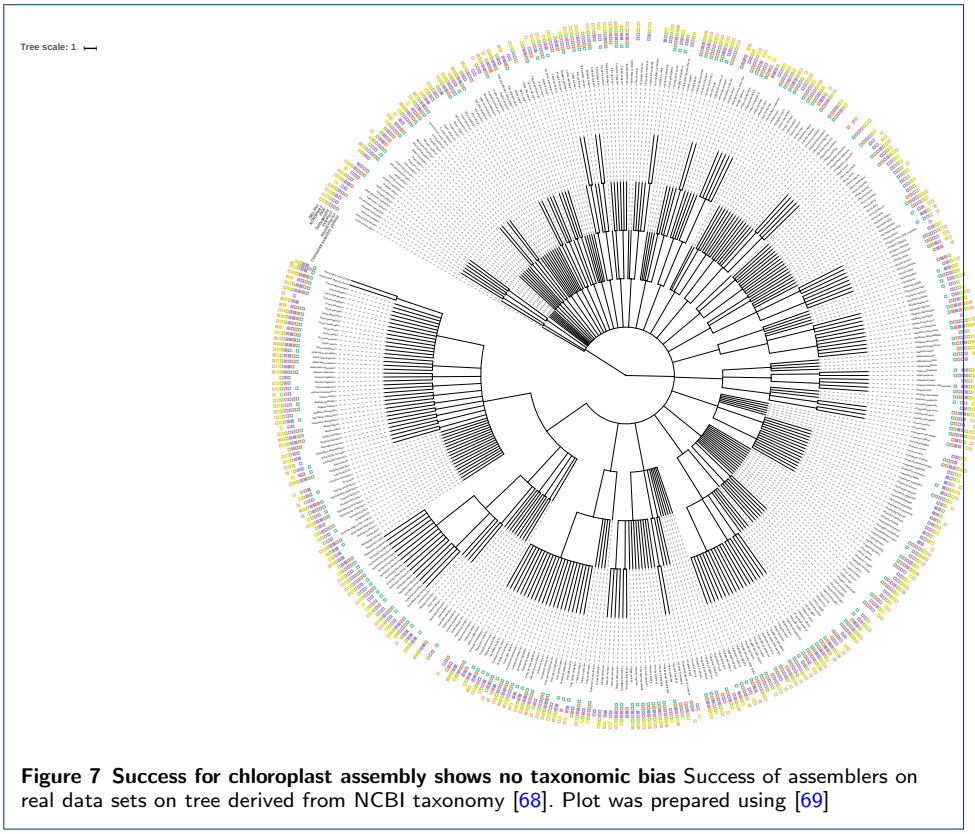

Figure 7 Success for chloroplast assembly shows no taxonomic bias Success of assemblers on real data sets on tree derived from NCBI taxonomy [68]. Plot was prepared using [69]