Apr 04, 2022

Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes

Kanae Nishii[1,2]

[1]Royal Botanic Garden Edinburgh; [2]Kanagawa University

1

dx.doi.org/10.17504/protocols.io.bx5dpq26

Kanae Nishii

Third generation NGS long-read sequences become popular for draft genome assembly. Here, we developed an efficient plant chloroplast assembly tool for whole genome assemblies from long-read datasets. This pipeline could be used also for mitochondrial genome assembly if the coverage is sufficiently high. The PLCL pipeline works on an assembled draft genome.

Summary of the PLCL pipeline

Step 1: Creating a blastn database of the draft genome assembly and search for target sequences
(chloroplast or mitochondria) as the queries.
Step 2: Checking the coverage (mean depth) of each contig in the draft genome using *minimap2* and *samtools*.
Step 3: Combining the results from step 1 ("blast_out.tsv") and step 2 ("coverage.tsv").
Step 4: *Optional*: *k*-means analyses for clustering the draft genome contigs with the Perl *Algorithm::KMeans* package.
Step 5: *Optional*: Visualizing *blastn* bitscore and *samtools*coverage data by R *ggplot2*.
Step 6: Selecting chloroplast contigs from the results of *blastn*, *samtools* coverage, and output of k-means analyses.
Step 7: Extracting the chloroplast / mitochondrial reads from the bam file created in step 2.
Step 8: Assembling reads into chloroplast / mitochondrial genomes.

DOI

dx.doi.org/10.17504/protocols.io.bx5dpq26

Kanae Nishii 2022. Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes. **protocols.io**
https://dx.doi.org/10.17504/protocols.io.bx5dpq26

protocol

Nishii K, Hart M, Kelso N, Barber S, Chen Y-Y, Thomson M, Trivedi U, Twyford A D, Möller M (2022) The first genome for the Cape Primrose *Streptocarpus rexii* (Gesneriaceae), a model plant for studying meristem-driven shoot diversity. *Plant Direct* (in prep)

blastn, contig coverage, mitochondrial genome assembly, Oxford Nanopore Technologies long-read sequencing, plant chloroplast genome assembly, samtools

protocol ,

Sep 10, 2021

Apr 04, 2022

53125

This pipeline worked for *Streptocarpus rexii* chloroplast and mitochondrial genome assembly with Oxford Nanopore Technologies (ONT) long-read sequencing datasets. The pipeline was also tested with *Arabidopsis thaliana* and *Orysa sativa* long-read sequencing datasets, and the chloroplast genomes of these model plants were successfully assembled.

This protocol shows the details of the PLCL chloroplast genome assembly using *Arabidopsis thaliana* "KBS-Mac-74" ONT reads (ERR2173373; Michael et al. 2018).

Programs involved

SRA Toolkit (https://trace.ncbi.nlm.nih.gov/Traces/sra)
NanoPlot (De Coster et al. 2018)
Quast (Gurevich et al. 2013)
Samtools (Li et al. 2009)
Minimap2 (Li 2018)
NCBI BLAST+ (https://blast.ncbi.nlm.nih.gov)
Perl Algorithm-KMeans-2.05 (https://metacpan.org/pod/Algorithm::KMeans)
R ggplot2 (Wickham 2016)
Wtdbg2 (Ruan and Li 2020)
Canu (Koren et al. 2017)

Relevant References

De Coster, W., D'Hert, S., Schultz, D.T., Cruts, M. and Van Broeckhoven, C. (2018) NanoPack: visualizing and processing long-read sequencing data. *Bioinformatics*, **34**, 2666-2669. [NanoPlot]

Gurevich, A., Saveliev, V., Vyahhi, N. and Tesler, G.,(2013) QUAST: quality assessment tool for genome assemblies. *Bioinformatics* **29,** 1072-1075. [quast]

Koren, S., Walenz, B.P., Berlin, K., Miller, J.R., Bergman, N.H. and Phillippy, A.M. (2017) Canu: scalable and accurate long-read assembly via adaptive *k*-mer weighting and repeat separation. *Genome Res.***27**, 722-736. [canu]

Li, H. (2018) Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics***34**, 3094-3100. [minimap2]

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R. and 1000 Genome Project Data Processing Subgroup (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics***25**, 2078-2079. [samtools]

Michael, T.P., Jupe, F., Bemm, F., Motley, S.T., Sandoval, J.P., Lanz, C., Loudet, O., Weigel, D. and Ecker, J.R. (2018) High contiguity *Arabidopsis thaliana* genome assembly with a single nanopore flow cell. *Nat. Commun.***9**, 541.

Ruan, J. and Li, H. (2020) Fast and accurate long-read assembly with wtdbg2. *Nat. Methods***17**, 155-158. [wtdbg2]

Wickham H. (2016) *ggplot2*: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org. [ggplot2]

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

Pre-starting PLCL pipeline

1 Download data for *Arabidopsis thaliana* from NCBI and check their read quality.

1.1 Download ONT reads

The data and information for *Arabidopsis thaliana*, ERR2173373 are available at https://www.ncbi.nlm.nih.gov/sra/?term=ERR2173373 and https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=ERR2173373
On the webpages, select the “Data access” tab on the top right to see the SRA information

sratoolkit.2.11.0-ubuntu64/bin/fastq-dump ERR2173373 –gzip

1.2 Download reference chloroplast sequence

Download reference chloroplast sequence for *Arabidopsis thaliana* “NC_000932” from
https://www.ncbi.nlm.nih.gov/nuccore/NC_000932

Click “FASTA” on the browser
From the drop down menu “FASTA”, select “FASTA(text)”
Copy and paste the sequence to a text editor
Save as “AtChloRef.fasta”

2 Quality checking ONT reads with *NanoPlot* and visualize read length distribution using R *ggplot2*

2.1 Checking the long-read quality with *NanoPlot*

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

a: *NanoPlot* standard output and automatic visualization

```
NanoPlot \
 -t 8 \
 --fastq $HOME/PLCL_At / ERR2173373.fastq.gz \
 -o OUTPUT_dir \
 --loglength
```

2.2 Checking the long-read quality with *NanoPlot*
b: *NanoPlot* raw output and manual visualization with R *ggplot2*

b-1: Store the raw data from *NanoPlot*

```
NanoPlot \
      -t 8 \
      --fastq $HOME/ONT_genome_raw.fastq.gz \
      -o OUTPUT_dir_raw \
      --raw \
      --store
```

b-2: R visualization of read length distribution as histogram

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

```
library(ggplot2)
df <- read.table("NanoPlot-data.tsv", header = T)
ONT <- ggplot(df, aes(x=lengths)) +
geom_histogram(bins=1000) +
  scale_x_log10(limits=c(1,1000000),labels=scales::comma)
+
  ylim(0,10000) +
  xlab("Read_length") +
  ylab("Number_of_reads")+
  theme(text=element_text(size=20))+
  geom_vline(xintercept = c(3500), linetype="dashed",
color="yellow")
ggsave("ONT.emf",plot=ONT)
```

R script

3 Assembling a draft genome for *Arabidopsis thaliana* from ONT reads using e.g. *wtdbg2* or *canu*

This example script uses *wtdbg2.*

3.1 Creating a lay file from raw reads

```
wtdbg2 \
      -x ont \
      -g 135m \
      -i $HOME/PLCL_At / ERR2173373.fastq.gz \
      -t 32 \
      -fo Atont
```

3.2 Generation of a draft genome assembly fasta file from a lay file

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

```
wtpoa-cns \
-t 32 \
-i ont.ctg.lay.gz \
-fo Atont.ctg.fasta
```

4 Optional: Assessing the draft genome assembly using *Quast*

```
quast \
 --large \
 -o Atont_wtdbg2_quast \
 -t 8 \
 Atont.ctg.fasta
```

PLCL pipeline: Step 1

5 Creating a *blastn* database of the draft genome assembly and search for target sequences (chloroplast or mitochondria) as the queries.

5.1 Generating a *blastn* database of draft genome assembly

```
makeblastdb \
      -in Atont.ctg.fasta \
      -out Atont.ctg_db \
      -dbtype nucl \
      -parse_seqids
```

5.2 Searching for chloroplast contigs using *blastn*

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

```
blastn \
      -db Atont.ctg_db \
      -query AtChloRef.fasta \
      -evalue 0.1 \
      -outfmt "6 sacc evalue bitscore" \
      -out blast_out.tsv
```

PLCL pipeline: Step 2

6 Checking the coverage (mean depth) of each contig in the draft genome using *minimap2* and *samtools*

6.1 Aligning raw ONT reads to the draft genome assembly using *minimap2*

```
minimap2 -ax map-ont Atont.ctg.fasta ERR2173373.fastq.gz
> ontasm.sam
```

6.2 Calculating coverage with *samtools*

```
samtools sort -o ontasm.bam -@ 8 ontasm.sam && \
samtools index ontasm.bam && \
samtools coverage -o coverage.tsv ontasm.bam
```

PLCL pipeline: Step 3

7 Combining the results from step 1 (“blast_out.tsv”) and step 2 (“coverage.tsv”)

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

7.1 In console, locate the “coverage.tsv” and “blast_out.tsv” files, and the “PLCL1.pl” script in the same folder and run:

```
Perl PLCL1.pl
```

PLCL1.pl

```
#!/usr/bin/env perl -w
use strict;
use warnings;

my $coverage="coverage.tsv";
my $blast="blast_out.tsv";
my $outfile="blast_cov.tsv";
my $outfile2="hitcontig_cov.tsv";
my $outfile3="hitcontig_cov.u.csv";
#step1 coverage data save as hash, key is rname
my %cov_data = ();
open COV,"<coverage.tsv" or die "!";
while(<COV>){
chomp;
my($rname,$start,$end,$numreads,$covbases,$coverage,$m
= split (/\t/, $_, 9);
$cov_data{$rname} =
[$start,$end,$numreads,$covbases,$coverage,$meandepth,$
}
close(COV);

#step2 merge blast data and meandepth
open OUT, '>', $outfile or die "!";
open OUT2, '>', $outfile2 or die "!";
open BLAST, '<',$blast or die "!";
while (my $line = <BLAST>){
chomp($line);
#$line =~ /^#/ and next;
my ($rname, $evalue, $bitscore, $length) = split(/\t/, $line, 4)
#search coverage ids
```

```
#search coverage ids
my $ref_data = $cov_data{$rname};
#add meandepth 1st place of cov_data
my $depth = $ref_data->[5];
#output
print OUT join ("\t",map $_ // "", $rname,$evalue,$bitscore,$le
print OUT2 join (",", map $_ // "", $rname,$depth), "\n";
}
close(OUT);
close(OUT2);
close(BLAST);

#step3 create table with only unique value
open IN, '<'. $outfile2 or die "!";
open OUT3, '>', $outfile3 or die "!";
my @array = <IN>;
my %count;
@array=grep{!$count{$_}++}@array;
print OUT3 @array, "\n";
close(IN);
close(OUT3);
```

PLCL pipeline: Step 4

8 *Optional: k*-means analyses for clustering the draft genome contigs with the Perl *Algorithm::KMeans* package

8.1 Locate “hitcontig_cov.u.csv” and “PLCL2.auto.pl” in the same folder and run:

```
Perl PLCL2.auto.pl 2>&1 > PLCL2.auto.log
```


Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

PLCL2.auto.pl

```
#!/usr/bin/env perl -w
use strict;
use warnings;
use Algorithm::KMeans;
#perlbrew switch perl-5.34.0
#perlbrew switch-off
#export LD_LIBRARY_PATH="/ gsl-2.6/lib"
#export LD_LIBRARY_PATH="/gsl-2.6/"
#/usr/sbin/setenforce 0

my $datafile="hitcontig_cov.u.csv";
my $mask="N1";
my $clusterer = Algorithm::KMeans -> new(datafile =>
$datafile,
                                mask => $mask,
                                K => 0,
                                cluster_seeding => 'smart',
                                terminal_output => 1,
                                write_clusters_to_files => 1,
                                );

$clusterer->read_data_from_file();

my ($clusters_hash,$cluster_centers_hash)=$clusterer-
>kmeans();
my $K_best=$clusterer->get_K_best();
my $QoCval=$clusterer->show_QoC_values();

my $outfile4="kmeans.out";
open OUT4, '>', $outfile4 or die "!";
print OUT4 "\ncluster\tcluster_center_value\tcontig\n";
foreach my $cluster_id (sort keys %{$clusters_hash}) {
   print OUT4 "\n$cluster_id\t@{$cluster_centers_hash->
{$cluster_id}}\t@{$clusters_hash->{$cluster_id}}\n";
}
close (OUT4);
```

Perl script for k-means analysis, automatic k selection

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

8.2 Locate “hitcontig_cov.u.csv” and “PLCL2.manual.pl” in the same folder and run:

```
Perl PLCL2.manual.pl 2>&1 > PLCL2.manual.log
```

PLCL2.manual.pl: Change [k] to desirable number of clusters

```
#!/usr/bin/env perl -w
use strict;
use warnings;
use Algorithm::KMeans;

my $datafile="hitcontig_cov.u.csv";
my $mask="N1";
my $clusterer = Algorithm::KMeans -> new(datafile =>
$datafile,
                                mask => $mask,
                                K => [k],
                                cluster_seeding => 'smart',
                                terminal_output => 1,
                                write_clusters_to_files => 1,
                                );

$clusterer->read_data_from_file();
my ($clusters_hash,$cluster_centers_hash)=$clusterer-
>kmeans();
my $K_best=$clusterer->get_K_best();
my $QoCval=$clusterer->show_QoC_values();

my $outfile4="kmeans.k[k].out";
open OUT4, '>', $outfile4 or die "!";
print OUT4 "\ncluster\tcluster_center_value\tcontig\n";
foreach my $cluster_id (sort keys %{$clusters_hash}) {
   print OUT4 "\n$cluster_id\t@{$cluster_centers_hash->
{$cluster_id}}\t@{$clusters_hash->{$cluster_id}}\n";
}
close (OUT4);
```

Perl script for k-means analysis, manual k selection

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

8.3 Combine the results of *k* means analyses to the "hitcontig_cov.u.csv" and save as "chlo_u.km.csv"

tig_ID	blastn		samtools	*k*-means analysis	
	evalue	bitscore	mean depth	*k = 0* (automatic *k*)	k = 5 (manual *k*)
tig_ID	evalue	bitscore	coverage	k0	k5
ctg70	0	37259	2151.53	c1	c4
ctg224	0	9075	3752.88	c1	c3
ctg225	0	5867	1752.46	c1	c4

Example result file: "chlo_u.km.csv"

PLCL pipeline: Step 5 (optional)

9 Visualizing *blastn* bitscore and *samtools* coverage data by R *ggplot2*

Plot the data "chlo_u.km.csv" from step 4 in R.

```
#load libraries
library(ggplot2)
library(dplyr)
library(tibble)
library(ggrepel)

#load chloroplast data
chlo <- read.csv("chlo_u.km.csv")
colnames(chlo)[5] <- "kmeans_k0"
colnames(chlo)[6] <- "kmeans_k5"
chlo$coverage <- as.numeric(chlo$coverage)
colnames(chlo)

#display k=0 plot without labels
ggplot(chlo) +
  aes(x=bitscore,y=coverage)+
```


Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

```
  geom_point(mapping = aes(color=kmeans_k0),size=4) +
  geom_point(colour="gray90",size=0.01)+
  scale_x_log10(limits=c(1,1000000),labels=scales::comma)+
  scale_y_log10(limits=c(0.1,1e4))+
  theme(text=element_text(size=20))

#display k=0 plot with labels > 50 coverage
ggplot(chlo) +
  aes(x=bitscore,y=coverage)+
  geom_point(mapping = aes(color=kmeans_k0),size=4) +
  geom_point(colour="gray90",size=0.01)+
  scale_x_log10(limits=c(1,1000000),labels=scales::comma)+
  scale_y_log10(limits=c(0.1,1e4))+
  theme(text=element_text(size=20))+
  geom_label_repel(data=chlo %>%
filter(coverage>50),aes(label=tig),nudge_x=0.1,nudge_y=0.2)

#save k = 0 plot with labels > 50 coverage
c0 <- ggplot(chlo) +
  aes(x=bitscore,y=coverage)+
  geom_point(mapping = aes(color=kmeans_k0),size=4) +
  geom_point(colour="gray90",size=0.01)+
  scale_x_log10(limits=c(1,1000000),labels=scales::comma)+
  scale_y_log10(limits=c(0.1,1e4))+
  theme(text=element_text(size=20))+
  geom_label_repel(data=chlo %>%
filter(coverage>50),aes(label=tig),nudge_x=0.1,nudge_y=0.2)
ggsave("At_chlo_k0.svg",c0)

#save k = 5 plot with labels > 50 coverage
c5 <- ggplot(chlo) +
  aes(x=bitscore,y=coverage)+
  geom_point(mapping = aes(color=kmeans_k5),size=4) +
  geom_point(colour="gray90",size=0.01)+
  scale_x_log10(limits=c(1,1000000),labels=scales::comma)+
  scale_y_log10(limits=c(0.1,1e4))+
  theme(text=element_text(size=20))+
  geom_label_repel(data=chlo %>%
filter(coverage>50),aes(label=tig),nudge_x=0.1,nudge_y=0.2)
ggsave("At_chlo_k5.svg",c5)
```

R script

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

PLCL pipeline: Step 6

10 Select chloroplast contigs from the results of *blastn*, *samtools* coverage, and *k*-means analyses.

Create a txt file of the list of contig name ("list.txt")

```
CTG121
CTG150
CTG1277
```

Example "list.txt" file

PLCL pipeline: Step 7

11 Extracting the chloroplast / mitochondrial reads from the bam file created in step 2

Locate "list.txt" in the working directory ($PWD) and run the commands "read_extract.sh".

read_extract.sh

```
mkdir $PWD/tempbam
for i in $(cat $PWD/list.txt);
do
	samtools view \
		-F 16 \
		-b \
		-o $PWD/tempbam/"$i".bam \
		$PWD/ontasm.bam \
		"$i"
done &&\
samtools merge $PWD/ontasm_filt.bam \
	-f \
	$PWD/tempbam/* && \
samtools fasta \
	$PWD/ontasm_filt.bam \
	-F 4 > \
	ontasm_filt.reads.fa
```


Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26

PLCL pipeline: Step 8

12 Assembling reads into chloroplast / mitochondrial genomes

```
canu \
-p Atont.chlo -d Atont_chloro_canu \
usegrid=false \
genomeSize=0.15m \
minReadLength=12000 \
minOverlapLength=10000 \
corMinCoverage=8 \
trimReadsOverlap=100 \
trimReadsCoverage=100 \
rawErrorRate=0.500 \
correctedErrorRate=0.144 \
corOutCoverage=40 \
minInputCoverage=8 \
stopOnLowCoverage=8 \
-nanopore-raw ontasm_filt.reads.fa
```

canu example script

Citation: Kanae Nishii Plant contig clustering based assembly (PLCL) pipeline: an efficient long-read assembly tool for plant chloroplast and mitochondrial genomes https://dx.doi.org/10.17504/protocols.io.bx5dpq26