# Using PACVr

Michael Gruenstaeudl, Nils Jenke

April 8, 2024

## Contents



## 1 Introduction

PACVr visualizes the coverage depth of a complete plastid genome as well as the equality of its inverted repeat regions in relation to the circular, quadripartite genome structure and the location of individual genes. This vignette provides instructions for generating the necessary input files and for executing the software from within the R interpreter via function `PACVr.complete()`, and invocation from the Unix command-line shell via script `PACVr_Rscript.R`. This vignette also illustrates the operation of PACVr on an empirical dataset co-supplied with the R package.

## 2 Installation

Prior to running PACVr, several dependencies need to be installed. Below is the most straightforward strategy in installing PACVr and all its dependencies on Linux (Arch Linux 4.18, Debian 9.9, and Ubuntu 18.10) and MacOS (HighSierra 10.13.6 and Mojave 10.14.6), respectively.
Open R and type:

```
if (!require("pacman"))
install.packages("pacman")
pacman::p_load("dplyr", "logger", "optparse", "read.gb", "RCircos", "BiocManager", install=TRUE)
BiocManager::install(c("BiocGenerics", "Biostrings", "GenomicAlignments", "GenomicRanges", "IRanges")
```

# 3 Running PACVr

## 3.1 Via R interpreter

PACVr can be executed from within the R interpreter via function `PACVr.complete()`.

```
library(PACVr)

# Specify input files
gbkFile <- system.file("extdata", "NC_045072/NC_045072.gb", package="PACVr")
bamFile <- system.file("extdata", "NC_045072/NC_045072_subsampled.bam",
                  package="PACVr")

# Specify output file
outFile <- paste(tempdir(), "/NC_045072_CoverageViz.pdf", sep="")

# Run PACVr
PACVr.complete(gbkFile, bamFile, windowSize=250,
           logScale=FALSE, threshold=0.5, syntenyLineType=1,
           relative=TRUE, textSize=0.5, regionsCheck=FALSE,
           tabularCovStats=FALSE, output=outFile)
```

## 3.2 Via Unix shell

PACVr can also be executed from the Unix command-line shell via script `PACVr_Rscript.R`.

```
Rscript ./inst/extdata/PACVr_Rscript.R \
 -k ./inst/extdata/NC_045072.gb \
 -b ./inst/extdata/NC_045072_subsampled.bam \
 -t 0.5 \
 -r TRUE \
 -o ./inst/extdata/NC_045072_CoverageViz.pdf
```

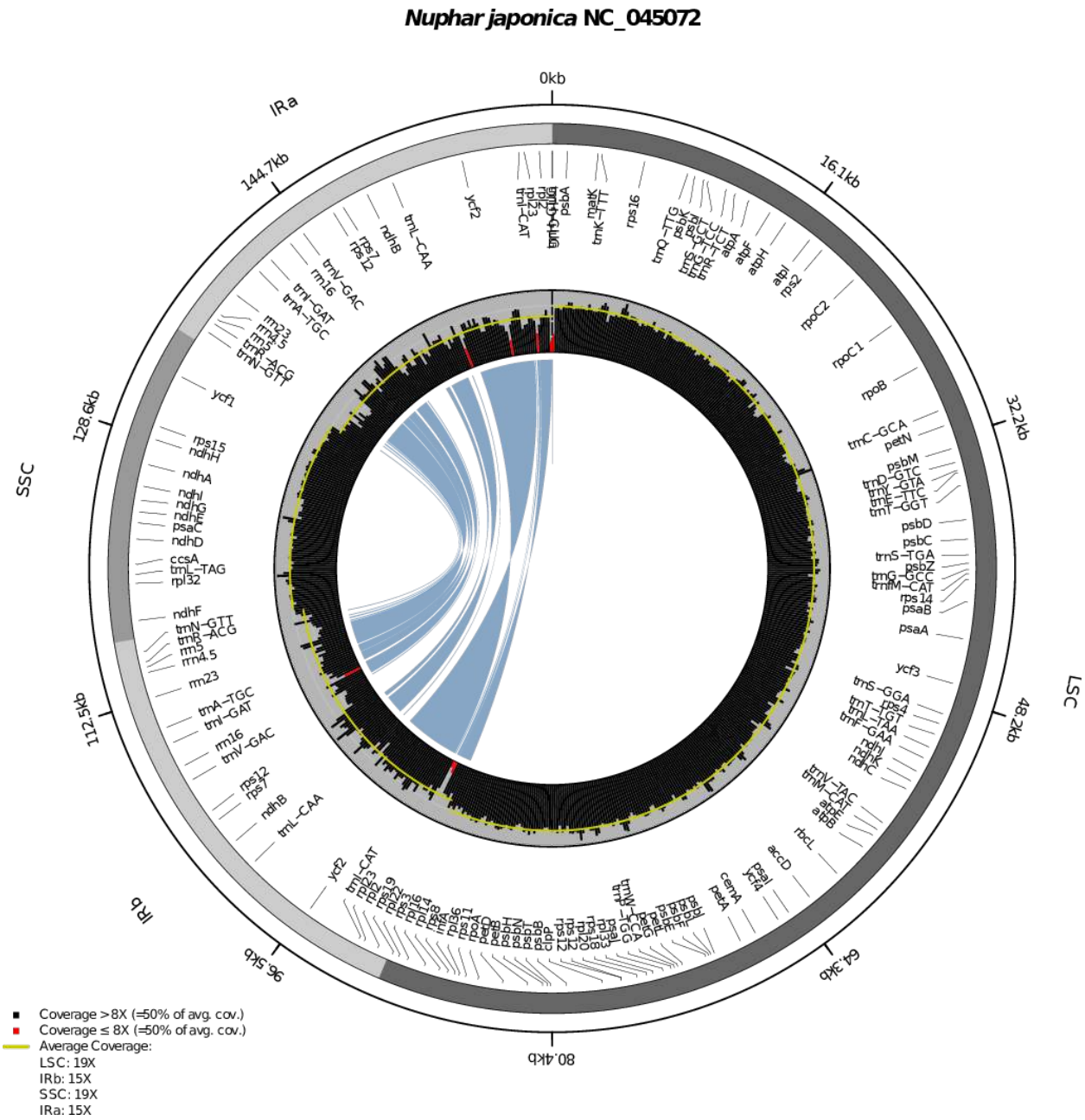


Figure 1: Coverage depth of the example dataset (*Nuphar japonica* NC_045072) as generated via `PACVr.complete()` when no filtering is conducted during read indexing via samtools.

# 4 Creating BAM file

For PACVr to compute and visualize coverage depth along a given plastid genome, users must provide textual information on the mapping of sequence reads on the input genome. This information is supplied via an input file in the binary alignment/map (BAM) format, which stores alignment and mapping information and is typically generated through the mapping of sequence reads to a reference genome with automated read-mappers in conjunction with the software samtools (Li 2009).

## 4.1 Option A: Mapping via BWA

The following is a minimal code example for mapping reads to a reference genome via BWA (Li and Durbin 2009).

```
# Building the index
bwa index <ref.fasta>

# Mapping reads to reference
bwa mem <ref.fasta> <reads_R1.fastq> <reads_R2.fastq> > <mapping_result.sam>
```

## 4.2 Option B: Mapping via Bowtie2

The following is a minimal code example for mapping reads to a reference genome via Bowtie2 (Langmead and Salzberg 2012).

```
# Building the index
mkdir -p db
bowtie2-build <ref.fasta> db/<ref_name>

# Mapping reads to reference
bowtie2 -x db/<ref_name> -1 <reads_R1.fastq> -2 <reads_R2.fastq> -S <mapping_result.sam>
```

## 4.3 (Optional – Indexing under different read filtering options)

The following are code examples for generating sorted and indexed BAM files via samtools under different read filtering options.

```
# Not filtered; 6877 reads in example dataset
samtools view -Sb <mapping_result.sam> > <all_reads.bam>

# Reads that map to one or more locations; 6854 reads in example dataset
samtools view -Sb -F 0x04 <mapping_result.sam> > <map_to_one_or_more_locations.bam>

# Reads that map to one location only; 6072 reads in example dataset
samtools view -S <mapping_result.sam> | grep -v "XA:Z\|SA:Z" |\
samtools view -bS -T <ref.fasta> - > <map_to_one_location_only.bam>

# Only reads that map to multiple locations; 2177 reads in example dataset
# Note: Ideal for confirming location of IRs!
samtools view -S <mapping_result.sam> | grep "XA:Z\|SA:Z" |\
```

```
samtools view -bS -T <ref.fasta> - > <must_map_to_multiple_locations.bam>

# Only reads that map in proper pairs; 28 reads in example dataset
samtools view -Sb -f 0x10 -f 0x20 <mapping_result.sam> > <map_in_proper_pairs.bam>

samtools sort <mapping_result.bam> > <mapping_result.sorted.bam>
samtools index <mapping_result.sorted.bam>
```

### 4.4 Output under different samtools filtering options

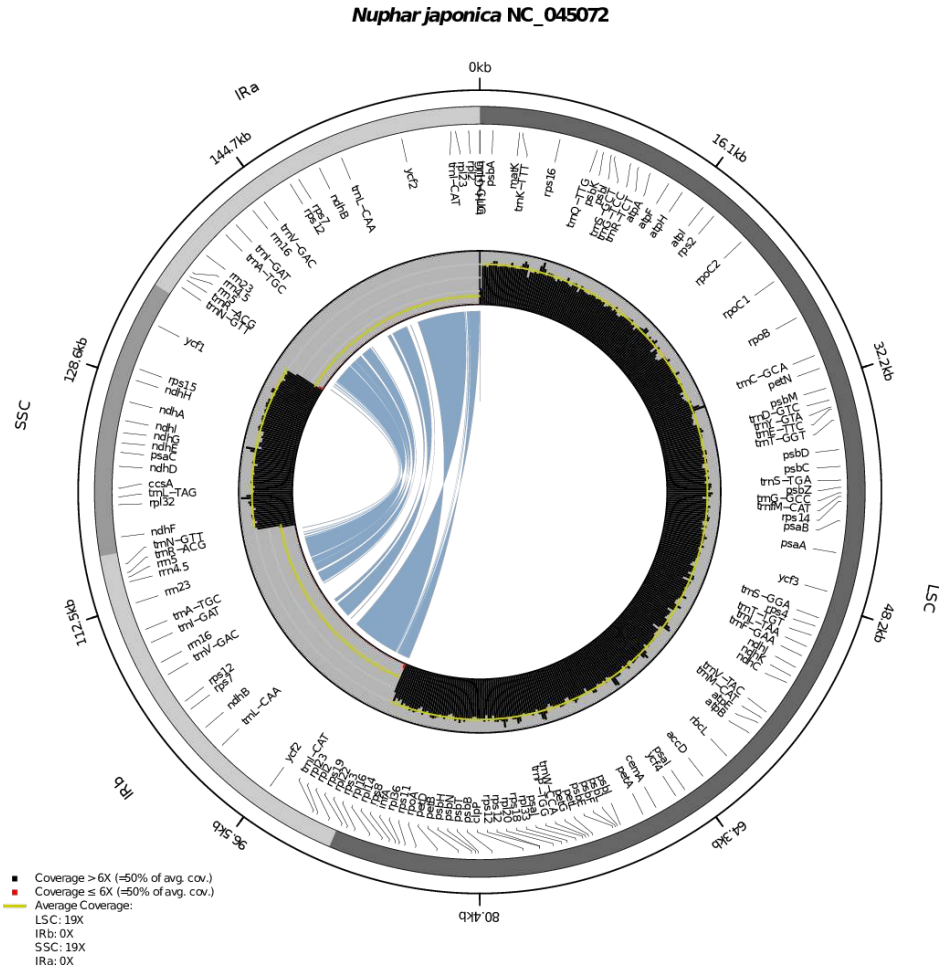


Figure 2: Coverage depth when only reads are used that map to one genome location.

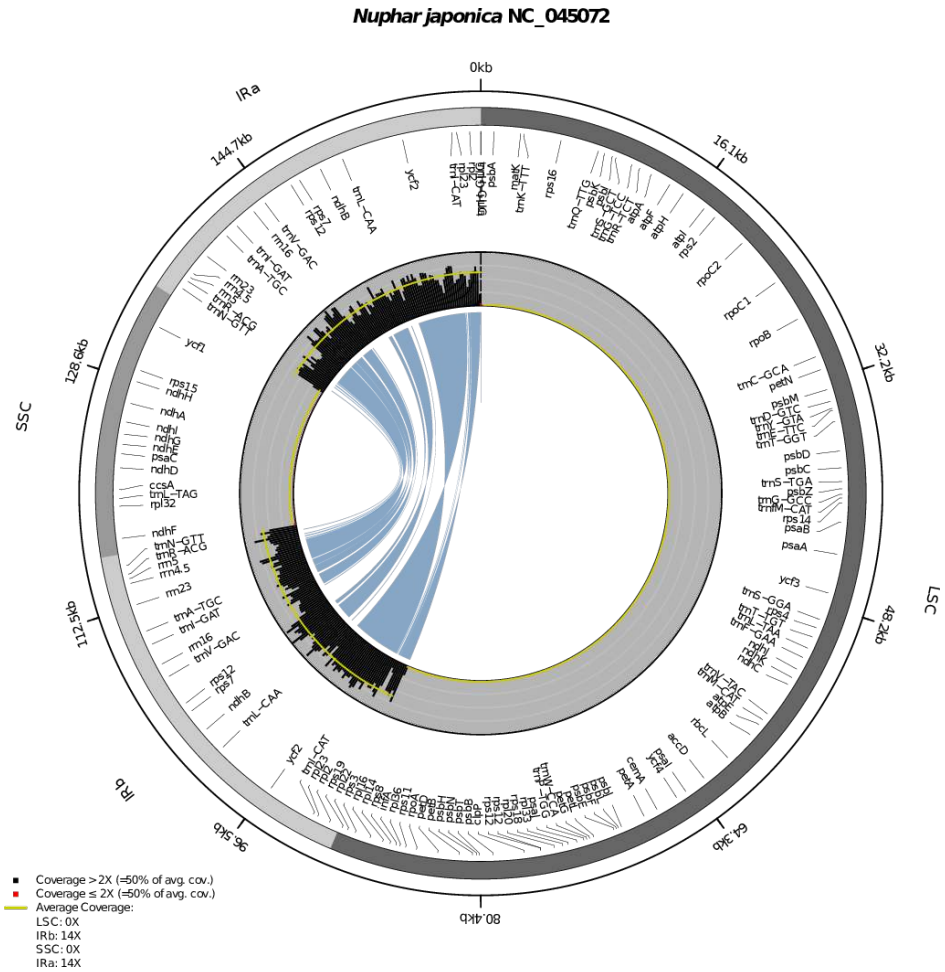


Figure 3: Coverage depth when only reads are used that to multiple genome locations; this setting is ideal to confirm the location of the IRs of a plastid genome.

## 4.5 (Optional – Subsampling reads to decrease the file size of the BAM files)

The following code helps to decrease the file size of BAM files by subsampling the input reads

```
samtools view -s 0.75 -b <input.bam> | samtools view -bS > <subsampled_input.bam>
```