nuccomp: rapidly summarize the contents of FASTA and FASTQ files containing nucleotides

Brian J. Knaus and Kelly J. Vining

2024, February 05

**To do:**

* Bioinformatics (impact factor: 6.931) - Application Notes
  + <https://academic.oup.com/bioinformatics/pages/instructions_for_authors>
  + up to 4 pages; this is approx. 2,600 words or 2,000 words plus one figure
* export to word?
* Cover letter
* 350 dpi tif figures; single (86 mm) or double (178 mm) column width
* second version of your figures at low-resolution for use in the review
* Applications notes: Title page, Short Structured Abstract, Text.
* References: The journal follows Oxford SCIMED style.

## Abstract

Bioinformatic projects include the management of large FASTA and FASTQ format files, such as a genomes. The contents of these files, and their quality, varies from project to project and file to file. For example, some genomes may include large numbers of ambiguous nucleotides which may present issues to other software used to analyze these files. This presents a need to rapidly determine the contents of these files. We present nuccomp as a tool to query and present summaries of the contents of FASTA and FASTQ files containing nucleotides. This tool consists of python scripts and an RMarkdown document. The python script can be run on local machines or high performance computing facilities and it rapidly summarizes the nucleotide composition for each sequence in each file. The RMarkdown provides an example of how this summary can be graphically presented as a report or as publication quality figures. With nuccomp we can rapidly determine the contents of FASTA and FASTQ format files so we can move on to downstream procedures with confidence in knowing the contents of these files.

Keywords: FASTA, FASTQ, genome announcement, nucleotide composition, quality control, sequence length distribution.

## Introduction

This paragraph is a great summary! This could be the first paragraph in the intro!

Genomic investigations typically begin with the management of FASTA and FASTQ format files, either produced by the investigator or aggregated from existing databases. The repository nuccomp consists of a suite of tools that utilizes the speed of python and Biopython (Cock *et al.* 2009) with the graphics and statistical resources of R (R Core Team 2020) combined with RMarkdown (Allaire *et al.* 2021) for report generation to summarize the contents of FASTA and FASTQ files containing nucleotides. The repository nuccomp provides a rapid solution to help inventory the contents of these files and presents this information in graphical and tabular forms.

Tools for summarizing bioinformatic files exist (Table 1), but they are currently distributed among many software packages designed to work on various file types. For example, the software FastQC (Andrews *et al.* 2010) summarizes raw sequencing reads in the FASTQ format. QUAST (Gurevich *et al.* 2013) is designed to evaluate genome assemblies by summarizing it and optionally aggregating summaries from other software. The trimmomatic (Bolger, Lohse and Usadel 2014) attempts to remove nucleotides from low quality Illumina sequencing data. In order to infer whether gene space has been well assembled BUSCO (Simão *et al.* 2015) queries for conserved genes from several databases throughout the phylogenetic tree of life. Blobtools (Laetsch and Blaxter 2017) uses GC content and read coverage to attempt to improve assemblies. Some of these tools are not specific to FASTA or FASTQ format data, or may be a part of a more computationally expensive analysis that may be undesirable to repeat if only a summary is desired. Each of these tools does a fantastic job at what it was designed for. We feel that a solution specific to data in the FASTA and FASTQ format that is simple, so that it scales to multiple genomes, and includes elements that are implemented in high level code, so that they are easily customizable, would aid the management of genomics projects.

Do we need a table of tools? Column 1: Software name. Column 2: Author.

**Table 1.** Existing software that describe the contents of FASTA/Q files.

| Name | Author |
| --- | --- |
| Blobtools | Laetsch and Blaxter 2017 |
| BUSCO | Simão et al. 2015 |
| FastQC | Andrews et al. 2010 |
| QUAST | Gurevich et al. 2013 |
| trimmomatic | Bolger, Lohse and Usadel 2014 |

Is nuccomp a ‘suite’ of tools? There are two python scripts (nuccomp.py, motif\_counter.py) which are called by an RMarkdown document. Perhaps this needs some sort of clarification?

We present nuccomp, a suite of tools intended to summarize the contents of FASTA and FASTQ format files that contain nucleotides (Table 2). These tools consist of python scripts that summarize FASTA and FASTQ files in tabular format that is easy for other software to import, and an RMarkdown script that numerically and graphically summarizes the tabular data. Together, these scripts provide a rapid way to summarize the contents of FASTA and FASTQ files and presents this information in publication quality graphics, as well as reports for internal use. The use of RMarkdown not only provides an example of how to present the data in a visual format, it also provides a working example for how the end user can customize its presentation.

## Methods

DNA? No RNA? Amino acids? These are different ‘alphabets’ that would need to be handled differently. However, RNA-Seq is ‘DNA’ sequencing, is that the question?

The GitHub repository nuccomp (Knaus 2023) contains two python scripts, an RMarkdown document (Table 2), example data, and example output. The python scripts takes a FASTA or FASTQ format file, consisting of DNA nucleotides, as input. The outputs from the python scripts are summarized graphically and presented as formatted tables using the RMarkdown document. The RMarkdown is compiled, or ‘knit’, to an html document suitable for internal reports. Output can also be compiled to other formats supported by RMarkdown (html, pdf, docx; Xie, Allaire and Grolemund (2018)) and ggplot (\*.png, \*.tiff, and other formats; Wickham (2016)).

**Table 2.** Scripts included in nuccomp, the language they are implemented in, and brief description of functionality.

| Script | Language | Purpose |
| --- | --- | --- |
| nuccomp.py | python/biopython | Summarize whole sequences |
| motif\_counter.py | python/biopython | Windowize sequences |
| nuccomp.Rmd | RMarkdown/R | Numeric and graphical presentation |

The python script nuccomp.py takes as input a FASTA or FASTQ format file, which may or may not be compressed with gzip, as input. The python (≥ 3.8.2) script uses biopython (≥ 1.79; Cock *et al.* (2009)) to read in the file and to collect the length, as well as the (case sensitive) number of nucleotides and IUPAC ambiguous nucleotides. This information is output to a comma delimited text file. This script may be run independently or may be called from the RMarkdown document.

Clarify that motif\_counter.py queries for a regex motif.

Add formula for scaling and centering.

The python script motif\_counter.py divides each sequence in a FASTA or FASTQ file into windows of user specified length and then counts the number of motifs found with a regular expression within each window. The python (≥ 3.8.2) script uses biopython (≥ 1.79; Cock *et al.* (2009)) to read in the file and custom python to divide each sequence into windows and count the number of times a regular expression is matched within each window. The default regular expression is ‘CG’, as an attempt to quantify methylation potential, but can be modified by users who are familiar with regular expressions. This count is converted to a rate by dividing each count by it’s window size. This count is also scaled by subtracting the minimum and dividing by the maximum (after subtracting the minimum) and inverting by taking one and subtracting the scaled count (see equation). Creating a rate (motifs per position) helps to accommodate the last window of each sequence which is expected to be smaller than the user specified window size. This scaled rate is reported in the BED format output as column 5, the ‘score’ (SAMtools 2024). The coordinates for each window, the count, and score are reported in a BED format file.

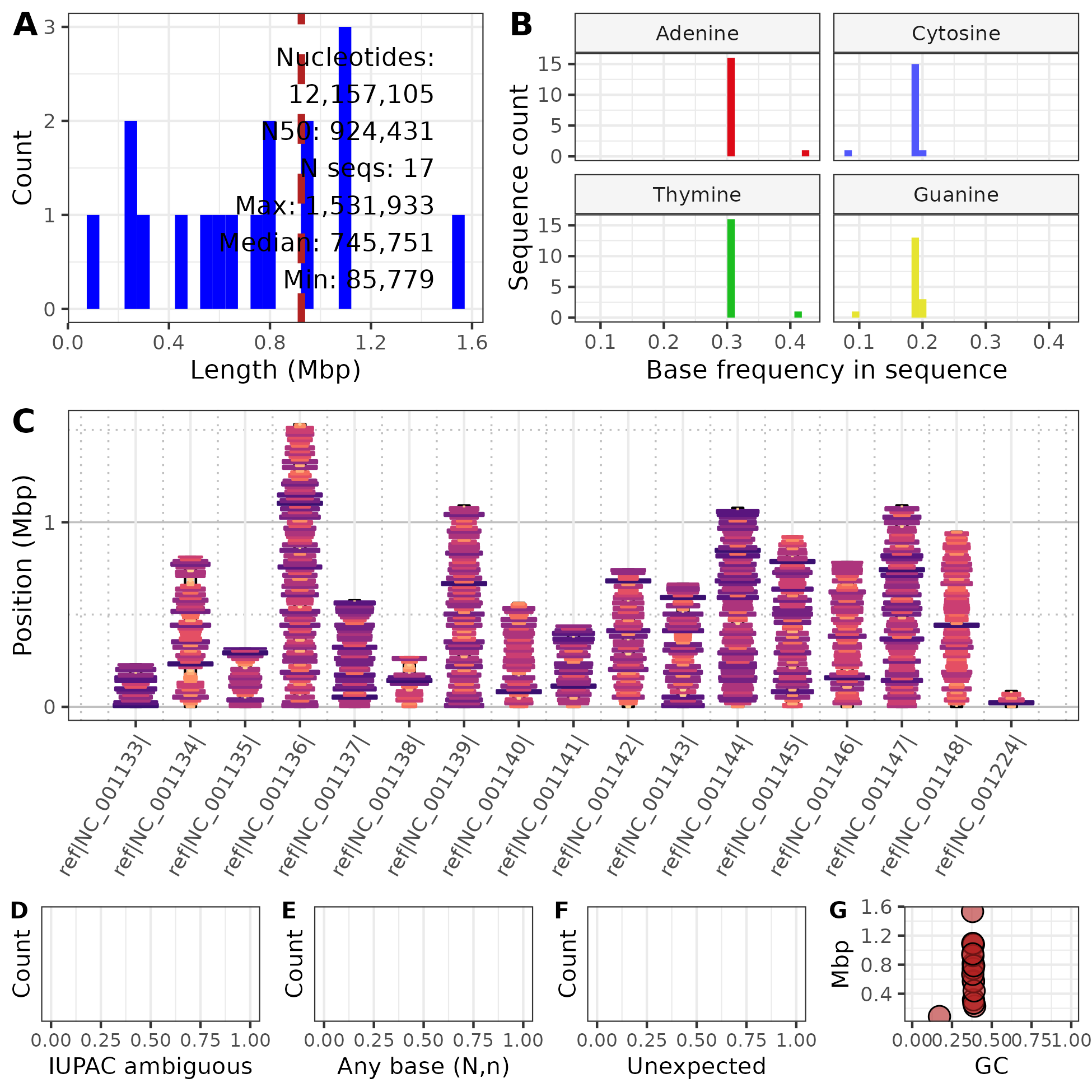
$$
scale = x-min(x) \\
scale = scale / max(scale) \\
scale = 1 - scale
$$

The RMarkdown (Xie, Allaire and Grolemund 2018) document uses R (R Core Team 2020) and reads in the comma delimited text file resulting from nuccomp.py and the BED file from motif\_counter.py to summarize them. Graphical summaries are made using ggplot2 (Wickham 2016) and ggpubr (Kassambara 2020) and consist of histograms of A, T, G, and C content, length of sequences, IUPAC ambiguity nucleotides (excluding A, C, G, T, and N), undetermined nucleotides (N), as well as a sanity check for unexpected characters. Numerical summaries are made as well such as the total number of nucleotides, N50, the number of sequences, and the median for these histograms. Examples are provided on the GitHub repository to demonstrate how to output these graphical summaries to \*.png format and publication quality \*.tiff format.

The *Saccharomyces cerevisiae* genome (<http://sgd-archive.yeastgenome.org/sequence/S288C_reference/genome_releases/>; Engel *et al.* (2014)) has been included as an example and a form of unit test. In addition, the potato (*Solanum tuberosum*; GCA\_001642375.1), hemp (*Cannabis sativa*; GCF\_900626175.2), mint (*Mentha longifolia*; GCA\_001642375.2), rice (*Oryza sativa*; GCF\_001433935.1) cottonwood (*Populus trichocarpa*; GCF\_000002775.5), hazelnut (*Corylus avellana*; GCA\_901000735.2), and *Arabidopsis thaliana* (GCF\_000001735.4) genomes were used for benchmarking. Benchmarking was performed on computer using an Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz CPU running Ubuntu 2022.04.2 LTS and R 4.3.0. The repository nuccomp is currently available on GitHub (Knaus 2023).

## Results

The python script nuccomp.py processed the 12 Mbp yeast genome, consisting of 17 sequences, in less than 1 second (large genomes consisting of more nucleotides and/or sequences should require more time). This resulted in a comma delimited text file consisting of 18 rows, a header describing the column contents, and a row for each of the 17 sequences in the S288C genome. This file included a column for the ‘ID’ of each sequence, it’s length, and the number of each nucleotide (A, C, G, T) as well as each ambiguous nucleotide all in a case sensitive manner (to accommodate masking). The python script “motif\_counter.py” summarized the yeast genome into windows into counts of the motif ‘CG’ and reported this to a BED format file. The RMarkdown script “nuccomp.Rmd” read in this these data, demonstrated the creation of individual plots from output of the python scripts, including a publication quality composite plot summarizing multiple data features. An example of this plot is presented in Figure 1. Execution of the RMarkdown script, including checks for dependencies, calls to the python scripts, and the rendering of graphics, required less than 10 seconds.



**Figure 1.** Summary of the yeast S288C assembly (R64-2-1\_20150113). Panel A is the distribution of sequence lengths. Panel B presents the distribution for per sequence, nucleotide frequencies. Panel C is an ideogram presenting the lengths (y-axis) of sequences. Each sequence is partitioned into 15 kbp windows for which the width and color were determined by the inverse scaled count of the motif ‘CG’ within each window. Panels D, E, and F summarize the occurrence of IUPAC ambiguous nucleotides (wWsSmMkKrRyY), any nucelotide (nN), and any unexpected nucleotides on a per sequence basis. Panel G presents the relationship between the length of each sequence and it’s G/C content.

Methods should be past tense???

In order to demonstrate the convenience of nuccomp, it was tested on several publicly available genomes. The potato (*Solanum tuberosum*) genome, with a size of 705 Mbp and an N50 of 1.3 Mbp, required 46.6 seconds. The hemp (*Cannabis sativa*) genome, consisting of 876 Mbp with an N50 of 91 Mbp, required 45.7 seconds. The mint (*Mentha longifolia*) genome, consisted of 468 Mbp and an N50 of 37 Mbp, required 29.0 seconds. The rice (*Oryza sativa*) genome, consisted of 374 Mbp and an N50 of 29 Mbp, required 25.1 seconds. The cottonwood (*Populus trichocarpa*) genome, consisted of 392 Mbp and an N50 of 19 Mbp, required 24.6 seconds. The hazelnut (*Corylus avellana*) genome, consisted of 369 Mbp and an N50 of 36 Mbp, required 23.7 seconds. The *Arabidopsis thaliana* genome, consisted of 119 Mbp and an N50 of 23 Mbp required 12.4 seconds. Processing time included the running of python code and compiling (‘knitting’) the RMarkdown, including text input and graphics rendering. Benchmarking results can be found at the GitHub repository (Knaus (2023)) in the directory ‘benchmarks’.

## Discussion

Current genome sequencing technology reports this DNA in a text file organized in the FASTA or FASTQ format (Cock *et al.* 2010). Genomics projects therefore involve managing FASTA and FASTQ files. Part of this management involves summarizing the contents of these large files in order to validate their contents as well as aid in project organization. This entails collection of summaries of the FASTA files from the various workflow outputs. A significant amount of project time may be invested in determining which summaries are available from which output and how to present this information.

The use of an RMarkdown document facilitates the summary of the data and also provides a high level code example for the summaries made. This RMarkdown document is a text file that is compiled, or ‘knit’, into several formats, such as \*html or \*pdf. The text file includes text as well as code chunks, as an attempt at literate programming. This means that the N50 summary statistic in the report includes the code used to generate the summary, providing transparency and educational value. Similarly, the graphical summaries include code examples for how the plots were generated. Because the summary table produced by nuccomp is saved as a text file, users have the potential to invent combinations of this information that the present authors haven’t imagined. It is hoped that providing this functionality will allow users to create their own innovations derived from this foundation.

The existing software QUAST (Gurevich *et al.* 2013) similarly summarizes assemblies by making it’s own summaries as well as aggregating information from other software. This makes it a feature rich option relative to nuccomp; however this richness comes with a performance cost. The simplicity of nuccomp lends itself to efficient execution time. This simplicity also lend itself to other applications. For example, we’ve begun using nuccomp.py to summarize long-read (PacBio) sequencing libraries after receiving them from sequencing centers and prior to assembly as a form of quality control. Users interested in a rich set of reports may enjoy QUAST, while users interested in simplicity and R code that they can modify may enjoy nuccomp. We hope many users will appreciate both options for their respective strengths. And we hope that users searching for a simple and customizable way of summarizing FASTA and FASTQ data will consider nuccomp.

## Acknowledgements

This work was supported by OregonCBD (Independence, Oregon, USA) through a gift to the Oregon State University Foundation.

## References

Allaire J, Xie Y, McPherson J *et al.* [*Rmarkdown: Dynamic Documents for R*](https://github.com/rstudio/rmarkdown). <https://github.com/rstudio/rmarkdown>, 2021.

Andrews S *et al.* [*FastQC: A Quality Control Tool for High Throughput Sequence Data*](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/). <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>; Babraham Bioinformatics, Babraham Institute, Cambridge, United Kingdom, 2010.

Bolger AM, Lohse M, Usadel B. [Trimmomatic: A flexible trimmer for illumina sequence data](https://doi.org/10.1093/bioinformatics/btu170). *Bioinformatics* 2014;**30**:2114–20.

Cock PJ, Antao T, Chang JT *et al.* [Biopython: Freely available python tools for computational molecular biology and bioinformatics](https://doi.org/10.1093/bioinformatics/btp163). *Bioinformatics* 2009;**25**:1422–3.

Cock PJ, Fields CJ, Goto N *et al.* [The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants]( https://doi.org/10.1093/nar/gkp1137). *Nucleic acids research* 2010;**38**:1767–71.

Engel SR, Dietrich FS, Fisk DG *et al.* The reference genome sequence of Saccharomyces cerevisiae: Then and now. *G3: Genes, Genomes, Genetics* 2014;**4**:389–98.

Gurevich A, Saveliev V, Vyahhi N *et al.* [QUAST: Quality assessment tool for genome assemblies](https://doi.org/10.1093/bioinformatics/btt086). *Bioinformatics* 2013;**29**:1072–5.

Kassambara A. [*Ggpubr: ’Ggplot2’ Based Publication Ready Plots*](https://CRAN.R-project.org/package=ggpubr)., 2020.

Knaus BJ. *Nuccomp*. <https://github.com/knausb/nuccomp>; GitHub, 2023.

Laetsch DR, Blaxter ML. [BlobTools: Interrogation of genome assemblies](https://doi.org/10.12688/f1000research.12232.1). *F1000Research* 2017;**6**:1287.

R Core Team. [*R: A Language and Environment for Statistical Computing*](https://www.R-project.org/). Vienna, Austria: R Foundation for Statistical Computing, 2020.

SAMtools. *BED V1*. <https://samtools.github.io/hts-specs/>; GitHub, 2024.

Simão FA, Waterhouse RM, Ioannidis P *et al.* [BUSCO: Assessing genome assembly and annotation completeness with single-copy orthologs](https://doi.org/10.1093/bioinformatics/btv351). *Bioinformatics* 2015;**31**:3210–2.

Wickham H. [*Ggplot2: Elegant Graphics for Data Analysis*](https://ggplot2.tidyverse.org). Springer-Verlag New York, 2016.

Xie Y, Allaire JJ, Grolemund G. [*R Markdown: The Definitive Guide*](https://bookdown.org/yihui/rmarkdown). Boca Raton, Florida: Chapman; Hall/CRC, 2018.