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

Brian J. Knaus  
Corresponding author  
[brian.knaus@oregonstate.edu](mailto:brian.knaus@oregonstate.edu)  
Department of Horticulture  
Oregon State University  
4017 Agriculture and Life Sciences Building  
Corvallis, OR 97331

and

Kelly J. Vining  
Department of Horticulture  
Oregon State University

## 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

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, QUAST (Gurevich *et al.* 2013) is designed to evaluate genome assemblies by summarizing it and optionally aggregating summaries from other software. 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.



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

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. Documents can also be compiled to html, pdf, or docx formats supported by RMarkdown (Xie, Allaire and Grolemund 2018) and graphics can be produced in png and tiff formats supported by R (Wickham 2016; R Core Team 2020).



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.

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 rate typically does not vary throughout a large portion of the total range. For example, windows may vary from 70 – 75% presence of the motif, which may appear as small amount of variation when presented graphically on a full range of 0-100%. Scaling the data focuses the presentation on the fraction that is variable instead of using the full possible range. Scaling is performed 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.

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.

Comparisons were made to similar existing software. Quast (Gurevich et al. 2013), SeqKit (Shen et al. 2024), Seqstats (<https://github.com/clwgg/seqstats>), and sequence-stats (<https://github.com/raymondkiu/sequence-stats>) were used to summarize the *A. thaliana* assembly (GCF\_000001735.4\_TAIR10.1\_genomic.fna.gz). Comparisons were performed on a MacBook Air with an Apple M1 processor running Sonoma 14.5 and are available in the ‘comparisons’ directory of the nuccomp GitHub repository.

Benchmarking of nuccomp was performed on several genomic assemblies to explore performance over different assembly size and composition. Assemblies for *Arabidopsis thaliana* (GCF\_000001735.4), cottonwood (*Populus trichocarpa*; GCF\_000002775.5), downy mildew of hop (GCA\_003991265.1), hazelnut (*Corylus avellana*; GCA\_901000735.2), hemp (*Cannabis sativa*; GCF\_900626175.2), mint (*Mentha longifolia*; GCA\_001642375.2), potato (*Solanum tuberosum*; GCA\_001642375.1), rice (*Oryza sativa*; GCF\_001433935.1), and yeast (*Saccharomyces cerevisiae*; GCF\_000146045.2\_R64\_genomic.fna.gz) were processed with a MacBook Air with an Apple M1 processor running Sonoma 14.5 and are available in the ‘benchmarks2’ directory of the nuccomp GitHub repository.

## Results

The python script nuccomp.py processed the 119 Mbp *Arabidopsis* genome, consisting of 7 sequences, in less than 3 seconds (Table 1; larger genomes consisting of more nucleotides and/or sequences are expected to require more time). This resulted in a comma delimited text file consisting of 8 rows, a header describing the column contents, and a row for each of the 7 sequences in the 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.

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 outputs DNA in a text-based FASTA or FASTQ file 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 various bioinformatic tools. A significant amount of project time may be invested in determining which summaries are available from different tools and how to present this information.

The use of nuccomp facilitates the summary of DNA data and also provides a high level code example for the summaries. 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 hemp genome (Figure 1) provides an example of how nuccomp can rapidly summarize a genomic assembly. Figure 1A presents the distribution of sequence lengths where long sequences (chromosomes) are rare and smaller sequences (unplaced contigs) are more frequent. Figure 1B presents the nucleotide compositions for the sequences and we can see a peak of A and T richness at a little over 0.3 which represents a genome wide compositions, but that many sequences vary from this value. Figure 1C presents a summary of the first 20 sequences in the assembly and allows us to see the lengths of these sequences and the abundance of the ‘CG’ motif, which may help identify transcriptionally active regions as well as centromeres. Note that note of the other compared software provides this perspective. Figure 1D summarizes the presence of IUPAC ambiguity characters and provides positive validation that these features were looked for and were absent. The tabular presentation in Table 1 validates that these features were not present at low levels that may have been easy to miss in the graphical presentation. Figure 1E reports that undetermined or ‘N’ positions were present, likely due to the scaffolding of this genome. This is important because summaries such as N50 or sequence length include ambiguous nucleotides in their calculation. Panel 1F validates that there are no unexpected characters thast may have been accidentally introduced. Panel 1G plots the length of each sequence by its GC content, as inspired by Blobtools (Laetsch and Blaxter 2017). This demonstrates that longer sequences (chromosomes) have a similar GC content while the unplaced sequences vary in GC content. This variation in GC content may be useful in identifying chloroplasts, mitochondria, or bacterial infection. This graphic can be saved as a single multipanel graphic and is incorporated into a text report using Rmarkdownm, a features absent from the other compared software,

The existing software QUAST (Gurevich *et al.* 2013) similarly makes 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. 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, in part, 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.

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](%20https://doi.org/10.1093/nar/gkp1137). *Nucleic acids research* 2010;**38**:1767–71.

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.

Shen, Wei, Botond Sipos, and Liuyang Zhao. 2024. SeqKit2: A Swiss Army Knife for Sequence and Alignment Processing. *iMeta* 2024, e191.

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.

**Table 1.** Comparison of software that summarizes FAST[AQ] data. All comparisons were performed on the Arabidopsis thaliana genome (GCF\_000001735.4\_TAIR10.1\_genomic.fna.gz). Example output for each software can be found in the ‘comparisons’ directory of the nuccomp GitHub repository. A summary of the Arabidopsis assembly can be found in the ‘benchmarks’ directory of the nuccomp GitHub repository.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Software** | **Compileda** | **FAST[AQ]** | **Gzip** | **Tabular**  **outputb** | **Sequence**  **summaryc** | **Graphics** | **Seconds** |
| nuccomp.py | No | FA,FQ | Yes | Yes | Yes | No | 2.704 |
| Quast | No | FA | Yes | Yes | No | No | 5.427 |
| SeqKit | No | FA,FQ | Yes | Yes | No | No | 0.758 |
| Seqstats | Yes | FA,FQ | Yes | No | No | No | 0.2 |
| sequence-stats | No | FA,FQ | No | Yes | Yes | No | 40.858 |
| nuccomp.Rmd | No | FA,FQ | Yes | Yes | Yes | Yes | 7.051 |
| Quast\_plots | No | FA | Yes | Yes | No | Yes | 5.855 |
| SeqKit\_watch | No | FA,FQ | Yes | No | No | Yes | 0.763 |

a Depends on an independent compiler.

b Tabular output to file or standard out with minimal processing.

c Provides summaries of individual sequences within a single file.

**Table 2.** Tabular summary of the hemp CBDRx (GCF\_900626175.2\_cs10) assembly produced by nuccomp.

|  |  |
| --- | --- |
| **Category** | **Value** |
| Total length (bp) | 876,147,649 |
| Number of sequences (count) | 221 |
| N50 (bp) | 91,913,879 |
| Median length (bp) | 47,123 |
| Maximum length (bp) | 104,987,320 |
| Tenth longest sequence (bp) | 61,561,104 |
| Total A,a,C,c,G,g,T,t (bp) | 736,579,359 |
| Total C,c,G,g (bp) | 246,856,385 |
| GC/ACGT (percentage) | 33.514 |
| GC/Total length (percentage) | 28.175 |
| Number of Nn | 13,968,290 |
| Number of IUPAC ambiguous: wWsSmMkKrRyY | 0 |
| Number of unexpected | 0 |
| Processing time (secs) | 25.553 |

A screenshot of a graph

Description automatically generated

**Figure 1.** Summary of the hemp CBDRx (GCF\_900626175.2\_cs10) assembly. **Panel A** summarizes the distribution of sequence lengths in the assembly. **Panel B** summarizes the per sequence distribution for the rate of each nucleotide (case insensitive). **Panel C** is an ideogram (cartoon) for the first sequences where the height of each bar represents the sequence length and the with and color for windows is based on the per window inverse abundance of the ‘CG’ motif. **Panels D, E, and F** summarize the occurrence of IUPAC ambiguous nucleotides (wWsSmMkKrRyY), any nucelotide (nN), and any unexpected nucleotides on a per sequence/contig basis. When these panels are empty they provide positive validation that these features were searched for but not found. **Panel G** presents the relationship between the length of each sequence/contig and it’s G/C content.