Oct 13, 2025

N. gonorrhoeae LIN code methodology scripts

eLife

DOI
https://dx.doi.org/10.17504/protocols.io.4r3l21beqg1y/v1

Anastasia Unitt[1]

[1]University of Oxford

Anastasia Unitt
University of Oxford

OPEN ACCESS

DOI: https://dx.doi.org/10.17504/protocols.io.4r3l21beqg1y/v1

External link: https://doi.org/10.7554/eLife.107758

Protocol Citation: Anastasia Unitt 2025. N. gonorrhoeae LIN code methodology scripts. **protocols.io** https://dx.doi.org/10.17504/protocols.io.4r3l21beqg1y/v1

Manuscript citation:
Unitt A, Krisna MA, Parfitt KM, Jolley KA, Maiden MC, Harrison OB Neisseria gonorrhoeae LIN codes provide a robust, multi-resolution lineage nomenclature. eLife 14(). doi: 10.7554/eLife.107758

Protocol status: Other
This protocol documents the scripts that were used in the development of the N. gonorrhoeae LIN code, in accompaniment to the linked research article.

Created: October 10, 2025

Last Modified: October 13, 2025

Protocol Integer ID: 229543

Keywords: oxford university biomedical research computing, brief script, computing cluster, program, linux shell, intensive process, methodology

Abstract

Research article: https://doi.org/10.7554/eLife.107758.1

Brief scripts used to run programs referred to in the methodology of the article are described below. Most computationally intensive processes were run through the Oxford University Biomedical Research Computing (BMRC) computing cluster, which uses a linux shell.

Prokka and PIRATE

1 A representative dataset of isolates was first collated from PubMLST. These genomes were exported in gff format, and then re-annotated using prokka https://github.com/tseemann/prokka:

```
for f in $(cat /representative_genomes_filenames.txt); do prokka --outdir ./${f} --cpus 48 --compliant --force --prefix ${f} /output/$f;
done
```

2 The output annotated gff format files were input into PIRATE, to characterize the core genome https://github.com/SionBayliss/PIRATE:

```
module load PIRATE/1.0.5

PIRATE -i /prokka_gffs -o /pirate -a -r -t 24
```

PIRATE output analysis

3 “Pirate gene families ordered.tsv” was the main output file used this analysis, as it describes each locus across the dataset analysed. The column headed "gene_family" lists each locus.

allele_name	gene_family	consensus_gene_name	consensus_product	threshold	alleles_at_maximum_threshold	number_genomes	average_dose	min_dose	max_dose	genomes_containing_fissions	ge
g01628_00006	g01628	NA	hypothetical protein	98	1	896	1	1	1	0	
g00758_00006	g00758	Hgd	2-(hydroxymethyl)glutarate dehydrogenase	98	1	896	1	1	1	266	
g00296_00006	g00296	hisB	Imidazoleglycerol-phosphate dehydratase	98	1	896	1	1	1	0	
g00113_00006	g00113	hisC2	Histidinol-phosphate aminotransferase 2	98	1	896	1	1	1	0	

The "number_genomes" column details how many genomes each locus was present in, which was used to calculate the percentage presence. Paralogs were excluded by selecting loci for which the "genomes_containing_duplication" column was 0.

Once a preliminary core gene list was defined from this output, the prokka/PIRATE gene names were associated with their corresponding PubMLST gene numbers by cross-checking their start/end positions between the prokka gffs and PubMLST exported gffs. This cross-checking was done using VLOOKUP in Excel.

Local LIN code

4 LIN code thresholds were tested using a local installation of LIN code, which takes a table of allelic profiles as the input file https://gitlab.pasteur.fr/BEBP/LINcoding:

```
#syntax:
# -i refers to a tab separated values file
# -t column numbers for the selected identifiers e.g. 1-2, or 1,
5.
# -l column numbers for chosen alleles e.g. 3- meaning 3 to all
others
# -b set of thresholds - percentage of alleles matching. Threshold
100 is obligatory.

python LINcoding.py -i LINdataset.txt -t 1-2 -l 3- -b
06,0.12,0.25,0.43,0.62,9.27,24.7,40.15,46.32,54.35,57.44,64.9,100
```

Phylogenetic trees

5 FastTree was used to construct approximate maximum-likelihood trees via the galaxy platform https://usegalaxy.eu/root?tool_id=fasttree.

For true maximum likelihood trees, RAxML https://github.com/stamatak/standard-RAxML was used as follows:

```
module load RAxML/8.2.12-gompi-2021b-hybrid-avx2

raxmlHPC -m GTRGAMMA -p 619283 -T 48 -N 2 -s 1000alignment.fasta -
n 1000tree.nwk
```

ClonalFrameML https://github.com/xavierdidelot/ClonalFrameML was then applied to correct these trees for the effect of recombination:

```
ClonalFrameML 1000tree.nwk 1000alignment.fasta output_prefix
```

Statistics

6 The Rand index was calculated in R using the package fossil **10.32614/CRAN.package.fossil**:

```
#Data should take the form of columns showing correlation between
clustering metrics e.g.
#column 1: isolate ID | column 2: metric #1 | column 3: metric 2

#convert to vectors
vector1<- as.vector(unlist(data$metric_1))
vector2<- as.vector(unlist(data$metric_2))

library(fossil)

rand.index(vector1, vector2)

adj.rand.index(vector1, vector2)
```

The Silhouette score was calculated using MSTclust, which was run as a java executable in a Linux environment https://gitlab.pasteur.fr/GIPhy/MSTclust:

```
#MSTclust takes a tab separated file of allelic profiles across a
set of
loci, and will calculate silhouette index for a give cutoff value
(-c)

java -jar MSTclust.jar -i unique_profiles.tsv -o
unique_profilesOUT_01 -l 1 -p 2- -c 0.01
```