Supplementary Methods

1. Selection of representative *Staphylococcus* genomes for TaxPhlAn target discovery

All genomes classified as *Staphylococcus* (n = 7247; 2016 Apr.) were downloaded from the NCBI assembled genomes database [1] using a custom in-house python script (ftp.ncbi.nlm.nih.gov/genomes/genbank/bacteria/) and from the NCBI Traces WGS database (https://www.ncbi.nlm.nih.gov/Traces/wgs/). The genomes with status (RefSeq category annotation) '*reference*' or '*representative*' and those with a genome length of 2-3Mb (2.8Mb on average) and with no more than 300 contigs / scaffolds were selected for analysis (n = 7085). The TaxPhlAn pipeline requires a sub-selection of input genomes due to computational capacity (n = 200-250 max). Therefore, we selected those genomes representing the largest degree of genomic differences in the *Staphylococcus* genus based on their 16S sequences as follows: from the filtered *Staphylococcus* genomes, 16S genes were determined by BLAST [2] with as query the full 16S gene of *S. aureus subsp. aureus 71193*. This sequence was downloaded from SILVA database [3] in April 2016; accession CP003045; this sequence was selected based on the longest available gene length (1831nt), in combination with the highest available sequence-, alignment- and pintail quality, which are 95.61%, 99.91% and 100.00%, respectively. As 16S genes are typically present in multi-copy in most bacterial genomes, for each genome only the hit with lowest E-value was evaluated. All genomes with a valid BLAST hit were considered (n = 7073), and for genomes with a minimum BLAST hit length of 1Kb the corresponding 16S region was used for further phylogenetic analysis (n = 6985). The selected 16S genes were aligned to the aforementioned 16S reference gene by a pairwise global alignment with Needle (version EMBOSS: 6.3.1; default settings) [4]. Based on these alignments, the 16S alignment region with the highest number of genomes covering at least 1Kb sequence was selected (n = 6569): this is position 296 to 1303 of the reference. 16S OTU clusters were determined using UCLUST (version 1.2.22q) [5] with percentage identity set to 99.7% (stepwords value set to 1007, in line with selected region length) which yielded 24 different clusters. From these 24 clusters we selected

200 unique *Staphylococcus* genomes in an automated manner, based on: (**i**) species annotation (i.e. taking along at least one genome per unique species for each cluster), (**ii**) genome status (i.e. reference / representative, closed / complete, or draft genome with ascending *n*-contigs or *n*-scaffolds, prioritizing in that order) as provided by the NCBI genome database [1], and (**iii**) with a minimum of one genome per cluster and a maximum of 10 genomes, for a total of 200 genomes.

2. Illumina sequencing data quality checking and demultiplexing

16S and SLST marker gene sequencing and collection of raw sequencing data was performed as detailed in the main text. The sequencing run was analyzed with the Illumina CASAVA pipeline (v1.8.3) with demultiplexing based on sample-specific barcodes. The raw sequencing data produced was processed removing the sequence reads of too low quality (only 'passing filter' reads were selected) and discarding reads containing adaptor sequences or failing PhiX Control with an in-house filtering protocol. A quality assessment on the remaining reads was performed using the FASTQC quality control tool version 0.10.0 [6].

3. TaxPhlAn (Module A): SLST target discovery and design workflow (see Supplementary Fig. S1)

3.1 Data preparation phase: Configuration file and input genomes. (Phase I-a)

In order to use the TaxPhlAn SLST design pipeline ('*TaxPhlAn-SLST-Design-wrapper.py*'), users are asked to provide a configuration file (i.e. *config.txt*) which should consist of the user-provided (private or public) genomes: detailing (**i**) their file name, (**ii**) genome identity (preferably strain-level annotation, if available), and (**iii**) status of the genome. Genome status can be '*ingroup*' for known ingroup genomes-of-interest, '*outgroup*' for a selection of known outgroup strains (representatives, preferably) and are used to identify SLST amplicon primers only recognizing the taxa in the ingroup, or '*extra*' for any remaining available genome of closely related stains and / or with unknown phylogenetic placement. We furthermore provide

the users the ability to assign '*strain-of-interest*' for one or multiple genomes that one wishes to explicitly distinguish from other input and '*strain-of-interest*' genomes. Note that a '*strain-of-interest'* is automatically treated as an ingroup genome, and the pipeline will provide information on candidates that best discriminate that genome from other genomes-of-interest. Please be advised that due to computational intensity it is not recommended for the total number of input genomes to exceed a maximum of 200-250 '*ingroup*' or '*strain-of-interest'* genomes (a larger number of genomes might work, but has not been extensively tested). The number of *'outgroup' or 'extra'* genomes may exceed this limit because these will be used for the (computationally less intensive) SLST target *in silico* PCR validation step only; nevertheless, we advise not to provide more than 500 of these additional genomes for similar reason. The input genomes can be provided as GenBank (compatible extensions are .gbk, .gbff or .gb) or as nucleotide FASTA files (extensions .fasta, .fas, .fa, .fsa, .fna or .seq are recognized), can be either single- or multi-contig (or scaffolds), and are allowed to have multiple files of the same type for each unique genome (GenBank or FASTA). The configuration file is mandatory, and has to contain all of the above information for each genome, in a tab-delimited format. A *config.txt* example file is provided in the pipeline upon download, including an example run on *Bifidobacterium* (both found in /TaxPhlAn/Pipeline-Design/). The location (path) of the input genomes directory and of the configuration file must be provided through argument values `--indir` and `--conf` on the command line, respectively. The provided configuration file will automatically be checked on required input information and format, and provided genome files are screened for any inconsistencies in file names as listed in the configuration file. Common user-input problems such as format inconsistencies or unmet criteria will be reported, and the user will be asked to specify, change, include or remove information accordingly. Providing an output directory is optional through the `--outdir` command, if not provided, output will be directed to ./TaxPhlAn.

3.2 Pre-processing phase: Input file reformatting and annotation of genomes. (Phase I-b)

In this phase, for each input genome, its contig(s) is extracted from its genome file(s), and if applicable, multiple contigs are concatenated into a single genome file and given a unique strain identifier (Strain1, Strain2, Strain3, etc.). In this process, a log-file of contigs (or scaffolds) joined to the same genome, and a translation table linking the new genome identifiers to their original contig(s) and genome will automatically be generated into (tab-delimited or FASTA-style, respectively) flat text files. After running properly, this pre-processing phase yields the user with cleaned-up, single-file standardized input genomes (in FASTA format, .fna) required for Prokka automated genome annotation [7]. In order to speed-up calculation time, annotation can be performed on multiple processor cores simultaneously (default number of cores is 1, maximum of 16 supported). For quality control (QC) purposes, a statistics file will be generated in the main output directory ('*statistics.txt*'), providing for each input genome information on its genome size, number of scaffolds / contigs, the GC-percentage and number of non-ATCG characters in the genome.

3.3 Orthology calculations and phylogenetic analysis. (Phase II)

For our proposed strain-typing method, it is key to determine single-copy, one-to-one orthologous gene clusters. The orthologous groups (OG) are calculated with orthAgogue [8] and requires the resulting output of an all-to-all protein BLAST (BLASTp; default settings) [9] as input data. These orthology calculations are based on a combination of the Inparanoid [10, 11] and Markov clustering (MCL) [12] algorithms underlying orthAgogue. In our TaxPhlAn pipeline, (Python-enabled) multiprocessing is supported for BLASTp, as well as for orthAgogue and Prokka; however, no more than 16 cores will be used by Prokka as internal tests have indicated that beyond that number computation time again increases by effect of supposed hard disk drive overhead processing (data not shown). The orthology matrix is reported as a tab-separated text file ('*matrix.txt'*, or '*matrix.annotated.txt*', in PhyloTree_OUT/ of main output

directory). For additional genome analysis, all single-copy one-to-one OG are determined and used for construction of a phylogenetic tree by: (**i**) alignment of the orthologous clusters (DNA-based alignments) by MUSCLE [13] (default settings), (**ii**) concatenation of all informative single nucleotide polymorphism (SNP) positions of each OG alignment into pseudo genes (i.e. one long stretch of SNPs / pseudo gene for each genome, based on all OG alignments), and finally, (**iii**) reconstruction of a phylogenetic tree based on the pseudo genes by FastTree [14] (default settings), which generates an iTOL-compatible newick tree file [15, 16] for visualization purposes, and a subsequent text-based phylogenetic tree by the command-line tool *newicktotxt* from NJplot [17, 18].

3.4 Selection of core gene clusters for candidate marker genes. (Phase III-a)

Based on orthology calculations by orthAgogue, single-copy OG (i.e. gene clusters; arbitrarily denoted by orthAgogue as cluster_1, cluster_2, cluster_3, etc.) are selected that meet the following criteria: (**i**) do not contain paralogs or multiple genes that are derived from the same genome, and (**ii**) are shared by at least 80% of the input genomes (default value, this setting can be adjusted). Importantly, as open reading frames (ORF) as predicted by Prokka are used for OG and gene cluster calculations, only gene-coding regions are considered; and consequently, inter-genetic regions which by their nature are not (strongly) subjected to evolutionary pressure (i.e. not typically carrying phylogenetically-linked sequence patterns) are dismissed. This process results in core clusters, and is reported by the pipeline as a tab-delimited text file.

3.5 Prediction of variable regions in candidate core clusters. (Phase III-b)

In the process of variable regions (VR) prediction, the TaxPhlAn algorithm starts by identifying highly conserved regions (CR) in the predicted core clusters, by a customizable set of rules. The core clusters are aligned with MUSCLE (default settings) [13]. Based on these cluster alignments, the algorithm builds-up

conserved regions from alignment start position to end. A conserved region is identified in an iterative fashion until a (penalty) limit is reached for any of the parameters as described in the next section. Hereafter, the conserved region is reported, and building of the next conserved region begins from the subsequent alignment position.

The algorithm for predicting variable regions computationally and iteratively evaluates all genetic regions within a certain range, in-between any combination of conserved regions pairs (i.e. conserved regions that have been identified based on specified settings at the first step of variable region detection). The default sequence span considered is 100-500nt, and this can be customized with the pipeline commands `--minlen` and `--maxlen`, for minimum and maximum variable region length, respectively. Note that, for this reason, one or more conserved regions can also be inside such a candidate variable region, but is nevertheless always flanked at its ends by a distinct conserved region pair suitable for SLST primer design. Consequently, any possible genetic region flanked by conserved sequences will be evaluated for its phylogenetic resolving power, unless it complies with the user-selected variable region length.

3.6 Penalizing system when searching for conserved (and variable) regions. (Phase III-c)

For finding conserved regions in cluster alignments, first, the TaxPhlAn algorithm will define a reference sequence based on the cluster alignment. For this reference sequence, the most frequent base at each alignment position is determined. In the common case of high base ambiguity on an alignment position, a degenerate IUPAC code base is allocated to the reference. A degenerate IUPAC base will be introduced to the reference sequence when 2, 3, or 4 different bases show >33%, >25% or >20% dominance at an alignment position, respectively (listed numbers are default values, and are customizable with commands `--n2`, `--n3` and `--n4`).

After the reference (consensus) sequence has been defined, the TaxPhlAn algorithm will iteratively build-up a conserved sequence by comparing all individual cluster sequences to the reference, again based on

the same cluster alignment. At each alignment position, the percentage of mismatches to the reference will be calculated, and if the reference sequence has a degenerate base at that position this will be recorded. If the percentage of mismatches is not higher than 20% (default), and if the number of degenerates in the already build-up conserved sequence does not exceed 4 (default), then the (degenerate) base at that position of the reference sequence is added to that conserved sequence (these two algorithm parameters are customizable with the pipeline commands `--mism` and `--deg`, respectively). This process is repeated at each cluster alignment position, until one of the (penalty) limits is exceeded, but also in case of IUPAC bases in any of the aligned sequences (not the reference), or when encountering alignment-introduced gaps. These unmet characteristics to further extend the already build-up conserved sequence are called 'failures' (set to 1 by default), and is customizable with command line option `--fails`.

When the maximum number of failures has been reached: (**i**) conserved sequence building stops immediately, and (**ii**) the currently build-up conserved sequence up to that alignment position is further evaluated. Thereafter, (**iii**) the conserved sequence is checked for its length, and is only reported if it meets the user-specified minimal conserved region length in order to facilitate primer design by Primer3 (15 by default, customizable with the pipeline command `--plen`) [19, 20]. Finally, (**iv**) all parameters are cleared from their penalties (i.e. reset to zero) and (**v**) conserved sequence building initiates again at the next alignment position.

3.7 in silico primer design and evaluation of variable (SLST) regions. (Phase IV)

The *in silico* predicted candidate variable regions are screened on a number of characteristics: a.o. (**i**) resolving power, (**ii**) size of the target region, (**iii**) number of genomes that contain the candidate region, (**iv**) correlation number to full-genome-based genome phylogeny and (**v**) number of primer sets available for that region. Thereafter, SLST candidates reported in the resulting output file are sorted first on

Spearman correlation to original full-genome phylogeny (maximum likelihood-based; ML), then on relative discrimination (defined as follows: (1 / (# input genomes + (# input genomes – # genomes recognized by primers of that candidate region, i.e. # hits)) * # unique VR sequences for that candidate region)). Primer sets of the top-level region candidates (100, by default) are selected and tested by PrimerProspector [21] in an *in silico* PCR on the full genome content of all input genomes (i.e. on all user-provided in- and outgroup genomes). For each primer set per candidate region, results (i.e. amplicons) from that *in silico* PCR by PrimerProspector are again analyzed on (**i**) number of amplicons identified (measure for number of genomes recognized by the primer pair), (**ii**) number of unique amplicons (in other words, resolving power of the region amplified), (**iii**) minimum and maximum amplicon size, and (**iv**) correlation value of amplicon-based phylogenetic relatedness to full-genome-based phylogeny.

Finally, top candidate regions and primers can be checked with Primer-BLAST to the NCBI's nt/nr database through a hyperlink provided by the pipeline, in order to check for undesired analogous gene regions and potential PCR-derived off-targets, respectively (found in SLST_OUT/ of the pipeline's main output directory: *Query-for-OFF-Targets-by-PrimerBLAST.html*) [22]. A final report named '*TaxPhlAn-candidates-REPORT.sorted.validated.txt'* is reported in this same directory. Alternatively, PDF files with a phylogenetic tree and alignment visualization of the top SLST candidates are generated based on their *in silico* performance, and can be found in the main output directory SLST_OUT/PPVal2-genomes/ (see for an impression Supplementary Fig. S3 for the main candidate of our *Bifidobacterium* example dataset).

4. TaxPhlAn (Module B): SLST data analysis (oligotyping) workflow (see Supplementary Fig. S2)

4.1 Data preparation phase: mandatory input and quality control. (Phase I)

An SLST oligotyping example run and test dataset is provided in the pipeline upon download (both found in /TaxPhlAn/Pipeline-Oligotyping/). To initiate, the main SLST data analysis wrapper script needs to be executed ('*TaxPhlAn-SLST-Oligotyping-wrapper.py*'), specifying the input directory with FASTA or FASTQ

files with the `--indir` option (mandatory input; reads should be assembled in case of paired-end sequencing), and optionally a user-specified output directory with the `--outdir` option. Furthermore, a mandatory reference alignment in FASTA format containing SLST reference sequences has to be supplied by the user with the `--fasta` option. This file (unique for each SLST candidate) is generated by the TaxPhlAn pipeline Module A, and can be found in SLST_OUT/PPVal2-genomes/ of the main output directory). Additional SLST references can manually be appended to the existing list (for example, references from other sources or databases, and any reference not taken along in the initial TaxPhlAn pipeline). Finally, the expected nucleotide length of the SLST sequences (i.e. alignment length, based on the references, primers included) has to be provided with the `--len` option (mandatory). Hereafter, reads are filtered on this expected length, only allowing reads within a three nucleotide (shorter) length of the reference SLST alignment to pass the filter. The read numbers for each sample are reported in an intermediate read statistics report on QC numbers and filtering: *'1_QC-report.txt'* in the main output directory.

4.2 Oligotyping by allele building. (Phase II)

Following the intermediate QC step, the pipeline run will be paused, thereby allowing for user-guided (manual) sample deselection for (suspected) spurious and / or low quality samples. This short user-interaction step will prevent use of erroneous SLST sequences for oligotyping; deselected samples are excluded in any further analysis. Only SLST reads passing above-described sequence length filter are aligned to the reference alignment by PyNAST, applying default settings [23]; reads longer than the reference alignment are discarded as they disrupt the reference alignment, which is not allowed by the PyNAST algorithm. Reads that are mapped by PyNAST to the reference alignment are retained for further analysis and appended to the existing reference alignment, thereby building an SLST reads super-alignment. In order to assign the informative SNP positions in the SLST sequences we determined the

Shannon diversity index (SDI) for each position in the SLST super-alignment, by adopting the Python module Shannon.py for calculating Shannon entropy (as published at GitHubGist: https://gist.github.com/audy/783125). Different SDI thresholds are evaluated (0.1 to 0.9, with steps of 0.1), and the number of informative SNP positions is calculated for each threshold, together with the corresponding number of unique alleles in the complete dataset. An 'allele' consists of the nucleotides present at the informative positions in a particular sequencing read. Note that output for a given SDI threshold is reported only if the number of obtained SNP positions is larger than 10; hence, this is similar to the length of the alleles (10 is the default value, this is customizable with command line argument `--minSNP`). We furthermore provide the option to exclude samples from these SDI calculations, by providing a list of samples with the `--list` option to be excluded for SDI calculations, but allele compositions of these samples will still be determined and reported. This can be relevant for positive- and negative control samples, or for instance when you have an imbalance in the number of (replication) samples for your individuals (hence, more replication samples of individual A compared to individual B could bias the SDI calculations towards individual A). Final output of this analysis phase is a list of all unique alleles found in the *entire dataset* for each SDI threshold, their overall relative abundances, and their corresponding reference genomes (if available); see flat-text file '*allele_info_complete_dataset-Th-0.*.txt*' in directory 5_Oligotyping_results/ (in tab-separated table format; sorted on the on average most abundant alleles). In the current study we used SDI thresholds of 0.6 for genus-level, and 0.2 for species-level, thresholds were empirically chosen based on work presented in this manuscript.

4.3 SLST sequences allele matching and scoring. (Phase III-a)

Each individual SLST read of the samples to be analyzed is converted to an allele depending on the different SDI threshold thresholds, and classified based on the known references, or classified as a (novel) unknown taxonomical entity (but with known phylogenetic placement). Note that this classification is performed without allowing mismatches in the allele sequences, hence only perfect hits are allowed.

4.4 Phylogenetic analysis of alleles, and data visualization. (Phase III-b)

For data output at each SDI threshold: annotation, data and phylogenetic tree files for use with iTOL [16] are automatically generated to accommodate the user with visualization of their results. In short, an allele-based phylogenetic tree figure of the top *N* most abundant alleles is generated as a PDF file, combined with a Bray-Curtis clustered heatmap of all SLST samples (underlying log10-transformed SLST relative abundances data). The individual iTOL-compatible files can be found in the directory 5_Oligotyping-results (and see 6_iTOL/ for intermediate files used for communication with the iTOL API at https://itol.embl.de/help.cgi#batch). Final visualization results for each SDI threshold are reported as '*iTOL-results-SDI0.*.pdf*' in the main output directory.

4.5 Oligotyping data results and reporting. (Phase IV)

In the final analysis phase, the number of reads corresponding to each unique allele is counted for every selected sample in the input directory, and is reported as an allele-to-sample (i.e. microbiota-to-sample, in those cases where an allele corresponds to a known reference) compositional matrix. Final output of these alleles *per sample* for each SDI threshold can be found in the directory 5_Oligotyping_results/ as flat-text files (in tab-separated table format) including allele absolute or relative abundances and taxonomical classifications: '*allele_Counts_per_Sample-Th-0.*.txt'* for absolute, and *'allele_RelAbund_per_Sample-Th-0.*.txt'* for relative abundances (sorted on the dataset-wide on average most abundant alleles). FASTA files with the *N* most abundant alleles in the dataset for each SDI threshold ('*alleles-selected-topN-Th-0.*.fasta'*) and the newick files thereof ('*alleles-selected-topN-Th-0.*.newick'*) can also be found in the directory 5_Oligotyping_results/. This top *N* is set to 100 by default, this value is customizable with `--topN` command-line argument. In addition, iTOL-compatible newick and annotation files to generate phylogenetic SLST trees (including allele alignments and heatmaps of SLST abundances,

etc.) can be found in the same directory. A final summary report is provided in the main working directory, providing the overall (and for each SDI threshold threshold) SLST read statistics: '*5_Summary-of-Results.txt*'. Furthermore, this report summarizes relevant information, a.o. with regard to number of informative SNP positions, number of alleles detected, number of reference genomes identified, and the dataset-wide overall relative abundance of the top *N* alleles.

5. TaxPhlAn pipeline dependencies : *Software*

Software, third party open source Programs and Tools, required to run TaxPhlAn :

name	version	reference
ARAGORN	1.2.38	[24]
BioPerl	1.7.2	[25]
BLAST+ *(blastn, blastp, makeblastdb)*	2.2.31+	[9]
CPAN	2.11	[26]
EMBOSS (*distmat, needle*)	6.6.0.0	[4]
FastTree (MP)	2.1.10	[14, 27]
FASTX Toolkit *(fastq_to_fasta)*	0.0.14	[28]
GNU Parallel	20180622	[29]
HMMER	3.1	[30, 31]
Infernal	1.1.2	[32]
Inparanoid	4.1	[10, 11]
iTOL 4 *(requires internet connection)*	web	[15, 16, 33, 34]
MCL	14-137	[12]
MUSCLE	3.8.31	[13]
Newick Utilities *(nw_reroot)*	1.6	[35]
newicktotxt	2.4.7	[17]
NumPy	1.14.5	[36, 37]
orthAgogue	1.0.1	[8]
PEAR	0.9.11	[38]
Perl	5.22.1	[39]
PerlBrew	0.84	[40]
PHYLIP *(neighbor)*	3.6	[41, 42]
Primer3	2.3.7	[19, 20, 43]
Primer-BLAST	web	[22]
PrimerProspector	1.01	[21]
Prodigal	2.6.3	[44]
Prokka	1.14	[7]
PyCogent	1.9	[45]
PyNAST	0.1	[23]
Python	2.7.15	[46]
R	3.3.3	[47]
SciPy	1.1.0	[48, 49]
tbl2asn	25.6	[50]
UCLUST	1.2.22	[5]

6. TaxPhlAn pipeline dependencies : *Libraries*

Python Libraries :

argparse
argv
collections
datetime
dateutil
dendropy
ete2
getopt
glob
inspect
math
matplotlib
multiprocessing
numpy
operator
os
pyparsing
random
re
scipy
scripts.shannon
scripts.textformatting
shlex
shutil
signal
string
stringio
subprocess
sys
time
traceback

Perl Libraries :

Bio::DB::Fasta
Bio::Perl
Bio::Seq
File::Basename
File::Spec::Functions
List::Util
Module::Build

References Supplementary Methods

1. Kitts, P.A., et al., *Assembly: a resource for assembled genomes at NCBI.* Nucleic Acids Res, 2016. **44**(D1): p. D73-80.
2. Altschul, S.F., et al., *Basic Local Alignment Search Tool.* Journal of Molecular Biology, 1990. **215**(3): p. 403-410.
3. Pruesse, E., et al., *SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB.* Nucleic Acids Research, 2007. **35**(21): p. 7188-7196.
4. Rice, P., I. Longden, and A. Bleasby, *EMBOSS: the European Molecular Biology Open Software Suite.* Trends Genet, 2000. **16**(6): p. 276-7.
5. Edgar, R.C., *Search and clustering orders of magnitude faster than BLAST.* Bioinformatics, 2010. **26**(19): p. 2460-1.
6. website. *FASTQC bioinformatics tool*. Available from: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/.
7. Seemann, T., *Prokka: rapid prokaryotic genome annotation.* Bioinformatics, 2014. **30**(14): p. 2068-9.
8. Ekseth, O.K., M. Kuiper, and V. Mironov, *orthAgogue: an agile tool for the rapid prediction of orthology relations.* Bioinformatics, 2014. **30**(5): p. 734-6.
9. Camacho, C., et al., *BLAST+: architecture and applications.* BMC Bioinformatics, 2009. **10**: p. 421-421.
10. Remm, M., C.E. Storm, and E.L. Sonnhammer, *Automatic clustering of orthologs and in-paralogs from pairwise species comparisons.* J Mol Biol, 2001. **314**(5): p. 1041-52.
11. O'Brien, K.P., M. Remm, and E.L. Sonnhammer, *Inparanoid: a comprehensive database of eukaryotic orthologs.* Nucleic Acids Res, 2005. **33**(Database issue): p. D476-80.
12. Enright, A.J., S. Van Dongen, and C.A. Ouzounis, *An efficient algorithm for large-scale detection of protein families.* Nucleic Acids Res, 2002. **30**(7): p. 1575-84.
13. Edgar, R.C., *MUSCLE: multiple sequence alignment with high accuracy and high throughput.* Nucleic Acids Res, 2004. **32**(5): p. 1792-7.
14. Price, M.N., P.S. Dehal, and A.P. Arkin, *FastTree: computing large minimum evolution trees with profiles instead of a distance matrix.* Mol Biol Evol, 2009. **26**(7): p. 1641-50.
15. website. *iTOL (interactive Tree of Life) bioinformatics tool*. Available from: http://itol.embl.de/.
16. Letunic, I. and P. Bork, *Interactive tree of life (iTOL) v3: an online tool for the display and annotation of phylogenetic and other trees.* Nucleic Acids Res, 2016. **44**(W1): p. W242-5.
17. website. *NJplot (newicktotxt) bioinformatics tool*. Available from: http://doua.prabi.fr/software/njplot.
18. Perrière, G. and M. Gouy, *WWW-query: An on-line retrieval system for biological sequence banks.* Biochimie, 1996. **78**(5): p. 364-369.
19. Koressaar, T. and M. Remm, *Enhancements and modifications of primer design program Primer3.* Bioinformatics, 2007. **23**(10): p. 1289-1291.
20. Untergasser, A., et al., *Primer3—new capabilities and interfaces.* Nucleic Acids Research, 2012. **40**(15): p. e115-e115.
21. Walters, W.A., et al., *PrimerProspector: de novo design and taxonomic analysis of barcoded polymerase chain reaction primers.* Bioinformatics, 2011. **27**(8): p. 1159-61.
22. Ye, J., et al., *Primer-BLAST: a tool to design target-specific primers for polymerase chain reaction.* BMC Bioinformatics, 2012. **13**: p. 134.

23. Caporaso, J.G., et al., *PyNAST: a flexible tool for aligning sequences to a template alignment.* Bioinformatics, 2010. **26**(2): p. 266-7.
24. Laslett, D. and B. Canback, *ARAGORN, a program to detect tRNA genes and tmRNA genes in nucleotide sequences.* Nucleic Acids Research, 2004. **32**(1): p. 11-16.
25. Stajich, J.E., et al., *The Bioperl toolkit: Perl modules for the life sciences.* Genome Res, 2002. **12**(10): p. 1611-8.
26. website. *CPAN, The Comprehensive Perl Archive Network*. Available from: http://www.cpan.org/.
27. Price, M.N., P.S. Dehal, and A.P. Arkin, *FastTree 2--approximately maximum-likelihood trees for large alignments.* PLoS One, 2010. **5**(3): p. e9490.
28. website. *FastX-Toolkit bioinformatics tool*. Available from: http://hannonlab.cshl.edu/fastx_toolkit/.
29. website. *GNU parallel - a shell tool for executing jobs in parallel using one or more computers*. Available from: http://www.gnu.org/software/parallel/.
30. website. *HMMER: biosequence analysis using profile hidden Markov models*. Available from: http://hmmer.org/.
31. Eddy, S.R., *Multiple alignment using hidden Markov models.* Proc Int Conf Intell Syst Mol Biol, 1995. **3**: p. 114-20.
32. Nawrocki, E.P. and S.R. Eddy, *Infernal 1.1: 100-fold faster RNA homology searches.* Bioinformatics, 2013. **29**(22): p. 2933-5.
33. Letunic, I. and P. Bork, *Interactive Tree Of Life (iTOL): an online tool for phylogenetic tree display and annotation.* Bioinformatics, 2007. **23**(1): p. 127-8.
34. Letunic, I. and P. Bork, *Interactive Tree Of Life v2: online annotation and display of phylogenetic trees made easy.* Nucleic Acids Res, 2011. **39**(Web Server issue): p. W475-8.
35. Junier, T. and E.M. Zdobnov, *The Newick utilities: high-throughput phylogenetic tree processing in the UNIX shell.* Bioinformatics, 2010. **26**(13): p. 1669-70.
36. website. *NumPy - for scientific computing with Python*. Available from: http://www.numpy.org/.
37. Oliphant, T.E., *Guide to NumPy.* USA: Trelgol Publishing, 2006.
38. Zhang, J., et al., *PEAR: a fast and accurate Illumina Paired-End reAd mergeR.* Bioinformatics, 2014. **30**(5): p. 614-20.
39. website. *The Perl Programming Language*. Available from: http://www.perl.org/.
40. website. *PerlBrew, an admin-free Perl installation management tool*. Available from: http://perlbrew.pl/.
41. website. *PHYLIP - a bioinformatics package of programs for inferring phylogenies*. Available from: http://evolution.genetics.washington.edu/phylip.html.
42. Baum, B.R., *PHYLIP: Phylogeny Inference Package. Version 3.2. Joel Felsenstein.* The Quarterly Review of Biology, 1989. **64**(4): p. 539-541.
43. website. *Primer3 - primer design program*. Available from: http://primer3.ut.ee/.
44. Hyatt, D., et al., *Prodigal: prokaryotic gene recognition and translation initiation site identification.* Bmc Bioinformatics, 2010. **11**: p. 119.
45. Knight, R., et al., *PyCogent: a toolkit for making sense from sequence.* Genome Biol, 2007. **8**(8): p. R171.
46. website. *The Python Programming Language*. Available from: http://www.python.org/.
47. website. *R: The R Project for Statistical Computing*. Available from: http://www.r-project.org/.
48. website. *SciPy - a Python-based ecosystem of open-source software for mathematics, science, and engineering.*; Available from: http://www.scipy.org/.
49. Oliphant, T.E., *Python for Scientific Computing.* Comput. Sci. Eng., 2007. **9:10–20**.

50. website. *tbl2asn - program that automates the creation of sequence records for submission to GenBank*. Available from: http://www.ncbi.nlm.nih.gov/genbank/tbl2asn2/.