PathogenSurveillance: an automated pipeline for population genomic analyses and pathogen identification

Zachary S. L. Foster (https://orcid.org/0000-0002-5075-0948)[1], Martha A. Sudermann (https://orcid.org/0000-0001-6912-3461)[2], Camilo Parada-Rojas (https://orcid.org/0000-0002-8372-6012)[2], Logan K. Blair[1], Fernanda Iruegas-Bocardo[2,3], Upasana Dhakal[2], Alexandra J Weisberg (https://orcid.org/0000-0002-0045-1368)[2], Hung Phan [2], Jeff H. Chang (https://orcid.org/0000-0002-1833-0695)[2], and Niklaus J. Grünwald (https://orcid.org/0000-0003-1656-7602)[1]

[1] Horticultural Crops Disease and Pest Management Research Unit, USDA Agricultural Research Service, Corvallis, OR, 97330
[2] Department of Botany and Plant Pathology, Oregon State University, Corvallis, Oregon 97331, USA
[3] Department of Plant Pathology and Environmental Microbiology, Pennsylvania State University, University Park, Pennsylvania U.S.A. 16802, USA

Abstract

Whole genome sequencing (WGS) offers a comprehensive, organism-agnostic method that effectively meets the need for efficient, reliable, and standardized responses to emerging threats from pathogens and pests. Here, we present *PathogenSurveillance*, an open-source and automated Nextflow pipeline for population genomic analyses of WGS data. It is designed with features tailored for biosurveillance and is suitable for in-field or point-of-care diagnostics. *PathogenSurveillance* is flexible, accommodating short- and long-read datasets and mixed samples of prokaryotes and/or eukaryotes. It automates all steps, including reference identification and retrieval from the NCBI Assembly database, and produces customizable interactive reports with summaries, phylogenetic trees, and minimum spanning networks that enable species and subspecies level identification. It also outputs quality control metrics and organizes genomic data hierarchically to facilitate downstream analyses. The pipeline runs on any Linux-based system and minimizes the need for advanced computational expertise. Source code is available on GitHub under the open-source MIT license. The pipeline expands the toolkit for real-time biosurveillance, enabling rapid detection and monitoring of pathogens and pests for rapid response to novel variants.

Interpretive summary

PathogenSurveillance is a new open-source tool that helps scientists quickly and reliably detect and monitor harmful organisms like pathogens and pests. It works by analyzing their genetic material, offering a fast and standardized way to respond to emerging biological threats. This tool is designed to be easy to use, even outside of traditional lab settings, such as in the field or at point-of-care locations. It can handle different types of genetic data, including those from bacteria, fungi, and other organisms, and works with both short and long DNA sequences. *PathogenSurveillance* automatically identifies reference genomes from public databases and generates interactive reports that include summaries, evolutionary trees, and network diagrams. These features help users identify organisms down to the species or subspecies level. It also

checks data quality and organizes results to support further analysis. Importantly, it runs on any Linux-based system and doesn't require advanced computing skills. The source code is freely available on GitHub under the MIT license, making it accessible to researchers and public health professionals worldwide. Overall, it adds a powerful tool to the biosurveillance toolkit, enabling faster responses to new and evolving biological threats.

Keywords: bioinformatics, biosurveillance, epidemics, epidemiology, Nextflow, One Health, pipeline, pandemic, pathogen genomics, population genomics

Introduction

Emerging invasive pathogens and pests pose an escalating global threat, largely driven by anthropogenic factors such as intensified international trade (Meyerson and Mooney 2007; Hulme 2009; Fisher et al. 2012; Liebhold et al. 2012). Their sudden appearance and potential to spread rapidly can outpace existing biosurveillance systems, making it difficult to mount timely responses (Carvajal-Yepes et al. 2019). Consequently, diseases caused by these novel organisms may reach epidemic levels before effective diagnostic methods are developed. To address this challenge, biosurveillance efforts require faster and more reliable detection that enables timely interventions (Gardy and Loman 2018; Hamelin et al. 2022). This is especially relevant for OneHealth frameworks, which emphasize the interconnected movement of pathogens across human, animal, and environmental domains, as seen in cases of antimicrobial resistance (Djordjevic et al. 2024).

Whole Genome Sequencing (WGS) is an important tool for biosurveillance, particularly in the early detection of infectious disease outbreaks (Weisberg et al. 2021). Methods for analyzing WGS data are highly effective for detecting emerging or reemerging invasive pathogens and pests, as they do not require *a priori* knowledge or optimization for specific organisms and can be applied to diagnosing rapidly developing threats even when the causative agent is unknown (Gardy and Loman 2018; Weisberg et al. 2021). However, key hurdles to the adoption of WGS are the need for appropriate infrastructure and domain expertise in both computational methods and pathogen biology (Oakeson et al. 2017; Iles et al. 2021). Thus, it is essential to design computational workflows that directly confront these challenges and empower users without specialized training to work with WGS data for disease detection.

WGS-based approaches commonly used for characterizing genetic clustering can be broadly grouped as *k*-mer, multigene, or variant-based phylogenies. *K*-mer based methods have the advantage of being extremely fast and computationally efficient (Pierce et al. 2019). They work by comparing the presence of only exact sequences of a given length (known as *k*-mers). The *k*-mer contents of genome sequences can be summarized by a systematically selected subset of hashed *k*-mers called a sketch and compared to a database of other sketches corresponding to reference genomes (Tian et al. 2020; Irber et al. 2022; Irber et al. 2024). However, sketch-based approaches are limited in their ability to infer taxonomic breadth, as their reliability declines with decreasing genome sequence similarity (Ondov et al. 2016). Conversely, multigene phylogenies can be used to group samples at various taxonomic levels, making such methods ideal for placing organisms of unknown identities in a phylogenetic context. But these approaches have limited resolution, are less scalable, and require domain knowledge to select

appropriate marker gene and reference sequences. The major strength of variant-based (single nucleotide polymorphisms (SNPs)) analyses lies in their high resolution, which enables clustering of closely related samples to identify clonal relationships and reveal potential transmission events. Variant-based approaches have similar limitations as multigene phylogeny-based approaches and are even more sensitive to reference selection and assembly quality (Weisberg et al. 2021).

Various pipelines have been developed for pathogen surveillance. *PathSeq* can be used to identify pathogens from host samples but relies on read alignment, thus requiring a reference database of potential microbes and host sequences (Walker et al. 2018). *Bactopia* and *Galaxy @Sciensano* are collections of seamlessly integrated tools, but they are designed for prokaryotes and require domain knowledge to select appropriate reference sequences (Petit and Read 2020a; Bogaerts et al. 2025). *CZ ID* (formerly *IDseq*) is a web-based pipeline for analyzing metagenomic data for pathogen detection but does not generate multigene phylogenies or high-resolution variant analyses (Kalantar et al. 2020). *Pathogenwatch* is a web service that relies largely on Multi-Locus Sequence Alignment (MLSA) based phylogenetic analyses but has utility for in-depth analyses (Argimón, David, et al. 2021; Argimón, Yeats, et al. 2021; Sánchez-Busó et al. 2021). However, *Pathogenwatch* is not optimized for use on local machines or private clouds and many of its tools are limited to specific pathogen groups. *PathoGFAIR* provides a Galaxy-based workflows to detect and track pathogens from metagenomic Nanopore sequencing (Nasr et al. 2025). Here we present nf-core/*PathogenSurveillance,* an open-source pipeline for automated population genomic and evolutionary analysis. This pipeline was released within nf-core, a collaborative, community-driven framework for developing and sharing standardized pipelines built on the Nextflow workflow management system (Di Tommaso et al. 2017).

PathogenSurveillance leverages nf-core modules and leverages community peer review to ensure robust functionality, while its use of Nextflow provides scalability, reproducibility, and portability. *PathogenSurveillance* automatically processes and analyzes short- and long-read WGS data along with publicly available genome assemblies. It features specialized capabilities for pathogen and pest identification, including flexible support for user-defined or automated reference sequence identification and retrieval, as well as the ability to concurrently analyze mixed prokaryotic and eukaryotic samples and populations. Other distinctive features include the integration of *k*-mer-, multigene phylogeny-, and variant-based approaches to deliver multiple levels of phylogenetic resolution for pathogen identification. The interactive output presents easily interpretable results, facilitating actionable responses. Moreover, output files are organized for deeper downstream investigation, extending use of *PathogenSurveillance* to population genomic studies to a wider range of organisms. To validate and illustrate its utility, we applied *PathogenSurveillance* to test datasets, demonstrating accuracy in automated reference selection and sample identification.

Implementation

We developed *PathogenSurveillance* for WGS data analyses of pathogens or pests. This pipeline is user-friendly and automates all steps from reference sequence selection and retrieval to the generation of interactive reports detailing sample identity, phylogenetic relationships, and population structure (Figure 1). The pipeline was designed according to widely accepted core

principles, with a particular focus on minimizing the need for advanced computational expertise by end users and incorporating specialized features for pathogen surveillance. *PathogenSurveillance* follows the FAIR (Findable, Accessible, Interoperable and Reusable) principles for processing data using workflows (De Visser et al. 2023).

PathogenSurveillance streamlines installation by leveraging Nextflow, an open-source software tool and domain-specific language that simplifies the creation and execution of data-intensive, scientific computational workflows (Di Tommaso et al. 2017). Nextflow uses container frameworks, such as Docker, Apptainer, or Conda, to automatically and reproducibly install all required dependencies as needed and upon first execution. Each process runs in its own isolated container and file system, eliminating software incompatibilities and ensuring consistent behavior across any Linux-based environment. As part of *PathogenSurveillance*'s development, we contributed seven new modules and made updates or bug fixes in eight existing nf-core modules (Supplementary Table 1).

The input process is also streamlined without sacrificing flexibility or configurability. *PathogenSurveillance* minimally requires a tab-separated value (TSV) file as input, with the only mandatory field being source of sequence data. This field can be populated with any combination of paths to local files or URLs to remote files, NCBI Short Read Archive (SRA) accession numbers, or NCBI Biosample IDs associated with SRA accessions. Input data may correspond to a single sample, multiple taxonomically disparate samples — including both prokaryotic and eukaryotic organisms —, populations of related samples, or any combination thereof. While *PathogenSurveillance* also has flexibility in analyzing samples with raw reads derived from Illumina, PacBio, or Oxford Nanopore technologies, it does not support data combined for the same sample from multiple sequencing platforms. For the latter case, users will need to select one source of read data. All other metadata for samples are optional but recommended, if available, to support downstream analyses. Users can also provide a second optional TSV file to specify reference genomes. Furthermore, users can indicate whether user-defined reference genomes should be included or used as the sole source, thereby excluding other possible references from analyses.

The pipeline is initiated with a single command. Upon initiation, it automatically generates a *k*-mer sketch from each sample's raw reads and compares these to sketches of all of NCBI RefSeq using bbmap sendsketch (Figure 1) (Bushnell 2014). If chosen by the user, it then identifies and retrieves assemblies that are possible reference sequences. Next, *PathogenSurveillance* processes and assembles sample reads and generates new *k*-mer sketches using sourmash to infer Average Nucleotide Identity (ANI) values that can be used to refine reference selection (Pierce et al. 2019). The pipeline selects the next workflow branches based on whether the sample is inferred to be of prokaryotic or eukaryotic origin. For prokaryotic datasets, it will annotate genome assemblies, generate ortholog clusters, identify shared orthologs among samples and their references, and construct a core genome phylogenetic tree. If the pipeline identifies clusters of closely related samples, it will map reads to their common reference assembly, call SNPs, and generate a SNP tree as well as a minimum spanning network. For eukaryotic datasets, *PathogenSurveillance* does not annotate genome sequences, as automated annotation is impractical because taxonomic groups require specialized training datasets. Instead, the pipeline generates a phylogenetic tree based on identified Benchmarking Universal Single-copy

Orthologs (BUSCO) sequences (Manni et al. 2021). A final interactive report is generated as an HTML file wherein users can define groups and adjust visualizations. All intermediate files, such as assemblies and annotations, are stored and accessible for additional analyses. *PathogenSurveillance* also produces quality control reports for reads and assemblies. Details on key workflow steps of the pipeline are provided in the following sections. The report created lists all the references or sources for computational tools used in the analysis.

PathogenSurveillance is portable and can be executed on any Linux-based platform, including local desktops, high performance computing (HPC) clusters, and commercial cloud environments. In addition, *PathogenSurveillance* can be launched from Sequera Platform, a web-based platform. A comprehensive guide to executing the pipeline, including input formats and optional parameters, is available online (https://nf-co.re/PathogenSurveillance).

Automated reference selection and population-level analyses

Effective selection of a reference genome and marker gene sequences are critical for accurate pathogen identification (Weisberg et al. 2021). However, the likely taxonomic classification of a sample may not be readily known, particularly for emerging pathogens and pests. To mitigate this limitation, we implemented a rule-based system for context-aware selection and retrieval of reference genome sequences that balances per-sample accuracy with the potential for coherent population-level analyses (Figure 2). By default, *PathogenSurveillance* attempts to automatically detect and group related samples within the analyzed dataset. Users also have the option of manually assigning references to samples or specifying which subgroups of samples should be analyzed together.

As described in the overview, rules are executed in two stages (Figure 1). In the first stage, a subset of possible reference assemblies relevant to the aggregated set of samples are downloaded (Figure 2A). For every unique family identified for samples analyzed, *PathogenSurveillance* will download the corresponding NCBI Assembly metadata, used to further refine reference selection based on genera, species, and strain information. If multiple references are available for a selected taxon, selection is additionally based on the prevalence of the associated taxon in the database indicating it is typical by NCBI standards, whether the assembly corresponds to a type strain, is in RefSeq, and associated with a Latin binomial, as well as assembly features such as assembly level, predicted assembly coverage of the genome sequence, absence of contaminating reads, and contig L50. In the second stage, the pipeline uses estimated ANI values to identify two types of reference assemblies among those downloaded (Figure 2). Contextual references are of well-characterized organisms selected to represent different taxonomic levels for phylogenetic analyses while mapping references are used for SNP calling.

To facilitate population-level analysis, the pipeline identifies contextual references that are shared across samples. This reduces the total number of distinct references and enables grouping of related samples for phylogenetic inference (Figure 2B). References are first partitioned into equal-sized bins based on a rescaled estimated ANI value for each sample. References most similar to each sample and those specified by the user are selected as initial candidates. *PathogenSurveillance* then iteratively selects additional references that best serve the entire group of samples. In each iteration, it prioritizes references that span the greatest number of bins

across all samples, recalculating bin coverage after each selection. This process continues until every bin is filled by at least one selected reference. This process also helps reduce runtime and computational resources needed to construct the phylogeny.

For prokaryotic samples, core gene sequences are used to construct a phylogeny while for eukaryotic samples, BUSCO gene sequences are used (Manni et al. 2021). If *PathogenSurveillance* determines that too few gene homologs are shared among samples, it partitions the dataset into the fewest possible number of clusters such that each cluster meets the minimum threshold of shared gene homologs required to support robust phylogenetic inference. The default threshold is set to 10 gene homologs.

Mapping reference assembly quality and phylogenetic distance from samples affects the accuracy of SNP calling after read mapping (Iruegas-Bocardo et al. 2023). Moreover, only samples whose reads are aligned to the same mapping reference assembly are directly comparable in population genomic analyses. *PathogenSurveillance* addresses these limitations by using estimated ANI values to cluster samples with their most closely related references in a sequential manner (Figure 2C). The process begins by selecting a reference that exceeds a defined minimum similarity threshold and is shared by the largest number of samples. When multiple references meet estimated ANI thresholds for the same number of samples, the one with the highest average similarity across those samples is chosen. This process is repeated iteratively until no samples can be matched to a reference. For each resulting cluster containing two or more samples assigned to the same reference, a pangenome graph method is executed to call SNPs (Eggertsson et al. 2019). This method is more robust to variation in phylogenetic distance and reference sequence quality (Eggertsson et al. 2019; Weisberg et al. 2021; Iruegas-Bocardo et al. 2023). SNP data is then used to create phylogenetic trees or minimum spanning networks.

***PathogenSurveillance* outputs**

PathogenSurveillance produces a rich set of data that are synthesized in tables and plots in interactive HTML reports for each user-defined group of samples. A sunburst plot provides a summary of taxonomic classifications of the samples, predicted based on initial *k*-mer sketches of raw reads and is associated with a table that presents the statistics for each match. These preliminary classifications, based on analysis of raw reads, serve to guide *PathogenSurveillance* in the next round for selecting reference assemblies. Next, estimated ANI and Percent of Conserved Proteins (POCP) values are displayed as both interactive heatmaps and in a table of best matches for each sample (Qin et al. 2014). These give essential context for interpretation of predicted taxonomic identities, as ANI reflects similarity across homologous regions while POCP captures overall genomic similarity between compared genome sequences (Weisberg et al. 2021). Multilocus, phylogenetic trees provide taxonomic placements of samples (Minh et al. 2020). A minimum spanning network constructed from genetic distances is used to visualize clonal lineages within populations (Kamvar et al. 2014). Table and plots are accompanied by expandable tooltips that explain how to interpret data and outline methods used.

In addition to the main report, *PathogenSurveillance* writes results from each step into structured folders that can be mined further for downstream analyses (Supplementary Figure 1). These output folders are organized hierarchically and named intuitively. Likewise, output files

corresponding to specific samples, references, or groups of samples and references are consistently named based on user-definable identifiers. The organization and naming are intentionally designed to be both human-readable and machine-parseable, facilitating seamless integration into downstream analyses. The kind of analyses used by *PathogenSurveillance* necessarily involve the creation of large and numerous files, not all of which are equally likely to be useful for downstream analysis. Nearly all intermediate files that take up little space are included in the output, but care has been taken to only include large files if their potential utility justifies their storage footprint. For example, quality filtering of BAM and VCF files occurs in several steps, each of which produces large files, but only the raw and final filtered versions of BAM and VCF files are preserved in the output. Additionally, by default all files in the output directory are links to files stored in a cache rather than the files themselves. This is the same cache that is used to resume pipeline runs and skip redundant steps. This has the benefit of only storing a single file when multiple runs of the pipeline produce the same file, as is common when testing multiple variations of a given dataset or pipeline parameters. Together, the careful curation of output files combined with the lack of redundancy significantly reduces the need for disk space, enabling larger analyses and efficient reanalysis of changing datasets.

The outputs also include quality control reports that support the interpretation of results. *PathogenSurveillance* automatically executes several standard quality control programs. FastQC is used to report basic and per read statistics for short read datasets while NanoPlot serves the same purpose for long read datasets (De Coster and Rademakers 2023). The adapter-trimming software, fastp, provides detailed reports on read filtering, trimming, and overall read quality (Chen et al. 2018). Quast is used to assess assembly quality metrics such as contiguity and completeness (Gurevich et al. 2013). BUSCO provides an additional estimate of genome assembly completeness by reporting the presence/absence of expected core genes. Finally, MultiQC is used to compile results from each of these programs into a single, comprehensive report, presenting quality control data for all samples in a consistent and well-organized format (Ewels et al. 2016).

Additional user-friendly features

PathogenSurveillance incorporates extensive parallelization, error recovery mechanisms, and caching of intermediate results, enabling improved efficiency and resilience in automated execution of large-scale, long-running analyses. Parallel processing occurs as system resources and API limits allow. In HPC clusters and commercial clouds, this can scale to hundreds or thousands of jobs running in parallel, each using all available central processing units (CPUs) on its assigned node or virtual machine. Processes that encounter errors are automatically retried a set number of times, with each attempt incrementally increasing Random Access Memory (RAM) and CPU allocation to accommodate resource demands. Processes that communicate with external web services will pause before retrying after a failed attempt, allowing time for temporary network issues to resolve. Information about pipeline execution, including any warnings or errors encountered during the process, are presented in output reports.
Caching of intermediate results by *PathogenSurveillance* offers three key advantages. First, if the pipeline terminates prematurely either due to an excessive number of failed processes or a user-issued command, it can be restarted from the point at which it stopped. This is made possible by using the "-resume" option. Second, caching gives users opportunities to incorporate

additional data or adjust parameters, without requiring extensive computing time. Only the affected portions of the analysis are recomputed and as a result, rerun executions are dramatically faster. Third, and similarly, caching enables users to iteratively refine evolutionary and population-level analyses. Users can first run *PathogenSurveillance* without specifying references, relying exclusively on automated selection, and then manually choosing more appropriate references after inspecting results. Reruns will be executed quickly and potentially be more robust. A drawback of caching is its large storage footprint, as seen in other comprehensive pipelines such as *Bactopia* (Petit and Read 2020b). We mitigated this to some extent by designing the pipeline to avoid storing redundant intermediate results.

Validation of PathogenSurveillance with published data

We evaluated published or custom datasets for accuracy and performance of *PathogenSurveillance*. The first dataset, used to test the accuracy of automated phylogenetic inference, is diverse and originally developed for a phylogenomic analysis of the *Serratia* genus (Williams et al. 2022). In previous work, authors used two methods to quality control sequence data and used two different methods to assemble long-read PacBio and short-read Illumina data. The pipeline was executed with the following command:

```
nextflow run nf-core/PathogenSurveillance -r 1.0.0 -resume -
profile docker,serratia_full --outdir serratia_results
```

Phylogenetic groups were further refined by using ANI clustering. Congruence between phylogenic trees was visualized using cophylogeny plots generated by the R package phytools (Revell 2024), ape (Paradis et al. 2004), phangorn (Schliep 2011), and ggtree (Yu et al. 2017).

From the original set of 664 samples, we manually selected a subset of 302 that captured the species and ecological diversity of the full set (Supplementary Table 2). The subset represented 11 species identified previously, with sequencing reads generated using long- and short-read technologies. We input their accession numbers and metadata into a TSV file. We also manually selected accessions of reference assemblies that were previously selected (Williams et al. 2022). The pipeline was executed with a single line command and in a mode that prevents *PathogenSurveillance* from automatically identifying and retrieving additional reference assemblies. *PathogenSurveillance* generated a phylogenetic tree, along with all the other associated intermediate assemblies, annotations, and ortholog cluster files as well as quality control metrics. The resulting tree was concordant in topology to the core gene phylogeny previously reported, and importantly, the species and lineage assignments were highly similar between the two analyses, with minor discrepancies restricted to placements within clonal lineages (Figure 3) (Williams et al. 2022). Overall, evaluation of this test dataset demonstrates that *PathogenSurveillance* can reliably generate phylogenetic and taxonomic insights with minimal user input.

Performance

Lastly, we generated prokaryotic and eukaryotic datasets to evaluate the impact of sample and genome sizes on runtime and RAM usage. One dataset was derived from a study of *Klebsiella*

pneumoniae, which has an average genome size of 5.7 Mb (David et al. 2020)(Supplementary Table 4). To evaluate the effect of sample size, we randomly selected samples to form runs of 1, 3, 5, 10, 25, 50, 75, 100, 150, and 200 samples. The pipeline was executed with caching disabled to analyze three replicates of each of the runs. As expected, runtime showed a linear increase from 0.4 to 11.7 hours as the number of samples increased from 1 to 200 samples (Figure 4A). RAM also increased linearly from 4.3 GB for 1 sample to 12.2 GB for 200 samples (Figure 4B).

We also created a dataset of taxonomically disparate organisms with genome sizes that vary from 2.1 Mb to 224.8 Mb (David et al. 2020)(Supplementary Table 5). Each dataset was composed of three samples of a single species and was run in triplicate. Each consists of Illumina paired-end reads, which were subset to 25x coverage for prokaryotes and 50x for eukaryotes. The results for prokaryotes and eukaryotes are displayed in different colors because they are processed by different workflow branches (Figure 1). Prokaryotes, regardless of genome size, were each completed in less than one hour of automated analyses (Figure 4C) with RAM usage less than 15.0Gb (Figure 4D). In contrast, the runtime for eukaryotic genomes had a substantial range that scaled linearly to genome size. The largest genome took 13.6 hours of runtime. RAM usage ranged between 6.0 GB and 70.0 GB for the eukaryotic dataset.

All tests were run on a System 76 Thelio Major Linux Desktop with a Ryzen 9 7950x 16-core processor.

Discussion

Pests and pathogens can emerge and rapidly adapt, often evading detection by traditional biosurveillance methods (Korber et al. 2020). In response to these challenges, we developed *PathogenSurveillance,* a user-friendly computational workflow for analyzing WGS data, with features specific for pest and pathogen detection. *PathogenSurveillance* provides a computational workflow that is highly portable for implementation on any Linux cloud or workstation. Our implementation follows the FAIR principles. Importantly, the pipeline can work with known or unknown taxa and can resolve clonality or novel variants if WGS data is available for the clades in question.

A key design principle of *PathogenSurveillance* is its commitment to lowering the barrier to entry and accelerating the analyses of WGS data. Population-level analysis requires domain specific knowledge in molecular evolution and bioinformatics, particularly for selecting appropriate references, installing software and dependencies, and constructing robust phylogenies. At its core, *PathogenSurveillance* is a generalizable population genomics pipeline, featuring a rules-based system that automates reference selection. Leveraging the Nextflow workflow management system, this design empowers users from diverse backgrounds, including those without formal training in molecular evolution and bioinformatics, to effectively analyze WGS data. Additionally, the organization of results reflects a deliberate design intended to support further in-depth investigations, both through manual and automated approaches. Collectively, *PathogenSurveillance* provides a streamlined and accessible framework for automating initial population genomic analysis, supporting both applied and fundamental research applications.

There are constraints to *PathogenSurveillance* that should be considered. First, the pipeline does not support analyses of viruses, as they would require their own workflow branch to parse metagenomic data and employ specialized approaches for taxonomic identification (Rangel-Pineros et al. 2023). Second, the accuracy in which the pipeline predicts taxonomic identities is constrained by the contents of databases used and the correctness of the taxonomic names they contain. Third, *PathogenSurveillance* was designed for WGS data derived from cultured organisms, which may limit applicability to metagenomic or environmental samples. Fourth, *PathogenSurveillance* employs default parameters across all steps, which may require appropriate adjustments for non-standard cases or refining reference selection. Finally, the pipeline can be computationally demanding. While the Nextflow framework allows efficient memory allocation per task, certain processes and samples may require substantial memory. Moreover, the pipeline generates numerous intermediate files, requiring storage capacity that is approximately five times greater than the size of input files.

Nonetheless, there are ways to apply *PathogenSurveillance* that mitigate these constraints. For example, we have identified contaminating reads in some sample datasets. When left unfiltered, the pipeline successfully resolved taxonomic identities and generated corresponding phylogenetic trees for both the primary and contaminating samples.. Regarding use of default parameters, caching gives users the flexibility of running the pipeline with a preliminary assessment and rerunning it with refined parameters or references without repeating the entire analysis. This not only saves considerable time but also supports reproducibility through automated, consistent execution. To mitigate storage requirements, users can clear the cache after analysis. However, for taxa that are repeatedly analyzed, retaining cached results may be advantageous, as it offsets rerun times.

In summary, *PathogenSurveillance* is executed with a single command line to automate the analyses of WGS data. It enhances traditional diagnostic approaches by improving analytical resolution and enriching the data available for more robust interpretation. It can also aid in identifying novel or emerging pathogens as well as place variants within taxa. These features underscore the potential of *PathogenSurveillance* in being a platform for consideration in global pathogen and pest biosurveillance (Carvajal-Yepes et al. 2019; Weisberg et al. 2021).

Data availability

PathogenSurveillance is free and open source under the MIT license. The source code is hosted on Github (https://github.com/nf-core/*PathogenSurveillance*) and documentation is available on nf-core (https://nf-co.re/*PathogenSurveillance*/). All data used in the validation of the pipeline presented here are available in the test data repository on GitHub (https://github.com/nf-core/test-datasets/tree/*PathogenSurveillance*). The scripts and input data used to create figures in this manuscript are also available on GitHub (https://github.com/grunwaldlab/*PathogenSurveillance*_publications).
Various test datasets automatically downloaded from NCBI of varying complexity available as profiles can easily be run as documented at https://github.com/nf-core/test-datasets/tree/*PathogenSurveillance*.

Acknowledgements

We thank members of the Chang and Grünwald laboratories for their valuable input and support. We also thank Melodie Putnam for her thoughtful advice on the design principles of *PathogenSurveillance,* Dr. Maude David for providing DNA from bacteria of the honeybee microbiota, and Charles Paulsen for extracting DNA and making libraries. We also thank the nf-core community for developing the infrastructure and resources for Nextflow pipelines as well as team, in particular Harshil Patel, Phil Ewels, Matthias Hörtenhuber, Mahesh Binzer-Panchal, and Maxime Garcia, for their assistance in reviewing and releasing the pipeline. Lastly, we thank the Center for Quantitative Life Sciences (CQLS) and Department of Botany and Plant Pathology (BPP) at Oregon State University and SCINet at the United States Department of Agriculture – Agricultural Research Service (USDA-ARS) for supporting high-performance computing infrastructures.

Funding

This work was supported by grants from USDA ARS (2072-22000-045-000-D) to NJG, USDA NIFA (2021-67021-34433; 2023-67013-39918) to JHC and NJG, USDA Plant Disease Recovery System (NPDRS) to NJG, and USDA APHIS to NJG.

Author Contributions

ZSLF designed and implemented nf-core/*PathogenSurveillance*. MAS, CPR, LKB, FIB, AJW, and HP contributed scripts and modules, and/or enhanced or tested pipeline reliability, usability and performance. ZSLF, MAS, CPR, JHC, and NJG wrote the manuscript, with input from all authors. NJG and JHC led the conceptualization and planning of nf-core/*PathogenSurveillance*, supervised its development, and secured funding.

References

Argimón S, David S, Underwood A, Abrudan M, Wheeler NE, Kekre M, Abudahab K, Yeats CA, Goater R, Taylor B, et al. 2021. Rapid Genomic characterization and global surveillance of *Klebsiella* using pathogenwatch. *Clinical Infectious Diseases* 73:S325–S335.

Argimón S, Yeats CA, Goater RJ, Abudahab K, Taylor B, Underwood A, Sánchez-Busó L, Wong VK, Dyson ZA, Nair S, et al. 2021. A global resource for genomic predictions of antimicrobial resistance and surveillance of *Salmonella typhi* at pathogenwatch. *Nat Commun* 12:2879.

Bogaerts B, Van Braekel J, Van Uffelen A, D'aes J, Godfroid M, Delcourt T, Kelchtermans M, Milis K, Goeders N, De Keersmaecker SCJ, et al. 2025. Galaxy @Sciensano: a comprehensive bioinformatics portal for genomics-based microbial typing, characterization, and outbreak detection. *BMC Genomics* 26:20.

Bushnell B. 2014. BBMap: a fast, accurate, splice-aware aligner. Available from: https://www.osti.gov/biblio/1241166

Carvajal-Yepes M, Cardwell K, Nelson A, Garrett KA, Giovani B, Saunders DGO, Kamoun S, Legg JP, Verdier V, Lessel J, et al. 2019. A global surveillance system for crop diseases. *Science* 364:1237–1239.

Chen S, Zhou Y, Chen Y, Gu J. 2018. fastp: an ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics* 34:i884–i890.

David S, Cohen V, Reuter S, Sheppard AE, Giani T, Parkhill J, the European Survey of Carbapenemase-Producing Enterobacteriaceae (EuSCAPE) Working Group, the ESCMID Study Group for Epidemiological Markers (ESGEM), Rossolini GM, Feil EJ, et al. 2020. Integrated chromosomal and plasmid sequence analyses reveal diverse modes of carbapenemase gene spread among *Klebsiella pneumoniae*. *Proc. Natl. Acad. Sci. U.S.A.* 117:25043–25054.

De Coster W, Rademakers R. 2023. NanoPack2: population-scale evaluation of long-read sequencing data.Alkan C, editor. *Bioinformatics* 39:btad311.

De Visser C, Johansson LF, Kulkarni P, Mei H, Neerincx P, Joeri Van Der Velde K, Horvatovich P, Van Gool AJ, Swertz MA, Hoen PAC 'T, et al. 2023. Ten quick tips for building FAIR workflows.Palagi PM, editor. *PLoS Comput Biol* 19:e1011369.

Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. 2017. Nextflow enables reproducible computational workflows. *Nat Biotechnol* 35:316–319.

Djordjevic SP, Jarocki VM, Seemann T, Cummins ML, Watt AE, Drigo B, Wyrsch ER, Reid CJ, Donner E, Howden BP. 2024. Genomic surveillance for antimicrobial resistance — a One Health perspective. *Nat Rev Genet* 25:142–157.

Eggertsson HP, Kristmundsdottir S, Beyter D, Jonsson H, Skuladottir A, Hardarson MT, Gudbjartsson DF, Stefansson K, Halldorsson BV, Melsted P. 2019. GraphTyper2 enables population-scale genotyping of structural variation using pangenome graphs. *Nat Commun* 10:5402.

Ewels P, Magnusson M, Lundin S, Käller M. 2016. MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics* 32:3047–3048.

Fisher MC, Henk DanielA, Briggs CJ, Brownstein JS, Madoff LC, McCraw SL, Gurr SJ. 2012. Emerging fungal threats to animal, plant and ecosystem health. *Nature* 484:186–194.

Gardy JL, Loman NJ. 2018. Towards a genomics-informed, real-time, global pathogen surveillance system. *Nat Rev Genet* 19:9–20.

Gurevich A, Saveliev V, Vyahhi N, Tesler G. 2013. QUAST: quality assessment tool for genome assemblies. *Bioinformatics* 29:1072–1075.

Hamelin RC, Bilodeau GJ, Heinzelmann R, Hrywkiw K, Capron A, Dort E, Dale AL, Giroux E, Kus S, Carleson NC, et al. 2022. Genomic biosurveillance detects a sexual hybrid in the sudden oak death pathogen. *Commun Biol* 5:477.

Hulme PE. 2009. Trade, transport and trouble: managing invasive species pathways in an era of globalization. *Journal of Applied Ecology* 46:10–18.

Iles LC, Fulladolsa AC, Smart A, Bonkowski J, Creswell T, Harmon CL, Hammerschmidt R, Hirch RR, Rodriguez Salamanca L. 2021. Everything is faster: How do land-grant university–based plant diagnostic laboratories keep up with a rapidly changing world? *Annu. Rev. Phytopathol.* 59:333–349.

Irber L, Pierce-Ward NT, Abuelanin M, Alexander H, Anant A, Barve K, Baumler C, Botvinnik O, Brooks P, Dsouza D, et al. 2024. sourmash v4: A multitool to quickly search, compare,and analyze genomic and metagenomic data sets. *JOSS* 9:6830.

Irber L, Pierce-Ward NT, Brown CT. 2022. Sourmash branchwater enables lightweight petabyte-scale sequence search. *bioRxiv*:2022.11.02.514947.

Iruegas-Bocardo F, Weisberg AJ, Riutta ER, Kilday K, Bonkowski JC, Creswell T, Daughtrey ML, Rane K, Grünwald NJ, Chang JH, et al. 2023. Whole genome sequencing-based tracing of a 2022 introduction and outbreak of *Xanthomonas hortorum* pv. *pelargonii*. *Phytopathology®* 113:975–984.

Kalantar KL, Carvalho T, De Bourcy CFA, Dimitrov B, Dingle G, Egger R, Han J, Holmes OB, Juan Y-F, King R, et al. 2020. IDseq—An open source cloud-based pipeline and analysis service for metagenomic pathogen detection and monitoring. *GigaScience* 9:giaa111.

Kamvar ZN, Tabima JF, Grünwald NJ. 2014. *Poppr*: an R package for genetic analysis of populations with clonal, partially clonal, and/or sexual reproduction. *PeerJ* 2:e281.

Korber B, Fischer WM, Gnanakaran S, Yoon H, Theiler J, Abfalterer W, Hengartner N, Giorgi EE, Bhattacharya T, Foley B, et al. 2020. Tracking Changes in SARS-CoV-2 Spike: Evidence that D614G Increases Infectivity of the COVID-19 virus. *Cell* 182:812-827.e19.

Liebhold AM, Brockerhoff EG, Garrett LJ, Parke JL, Britton KO. 2012. Live plant imports: the major pathway for forest insect and pathogen invasions of the US. *Frontiers in Ecology and the Environment* 10:135–143.

Manni M, Berkeley MR, Seppey M, Simão FA, Zdobnov EM. 2021. BUSCO Update: Novel and Streamlined Workflows along with Broader and Deeper Phylogenetic Coverage for Scoring of Eukaryotic, Prokaryotic, and Viral Genomes.Kelley J, editor. *Molecular Biology and Evolution* 38:4647–4654.

Meyerson LA, Mooney HA. 2007. Invasive alien species in an era of globalization. *Frontiers in Ecology and the Environment* 5:199–208.

Nasr E, Henger A, Grüning B, Zierep P, Batut B. 2025. PathoGFAIR: a collection of FAIR and adaptable (meta)genomics workflows for (foodborne) pathogens detection and tracking. *GigaScience* 14:giaf017.

Oakeson KF, Wagner JM, Mendenhall M, Rohrwasser A, Atkinson-Dunn R. 2017. Bioinformatic analyses of whole-genome sequence data in a public health laboratory. *Emerg. Infect. Dis.* 23:1441–1445.

Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, Phillippy AM. 2016. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol* 17:132.

Paradis E, Claude J, Strimmer K. 2004. APE: Analyses of Phylogenetics and Evolution in R language. *Bioinformatics* 20:289–290.

Petit RA, Read TD. 2020a. Bactopia: a Flexible Pipeline for Complete Analysis of Bacterial Genomes.Segata N, editor. *mSystems* 5:e00190-20.

Petit RA, Read TD. 2020b. Bactopia: a Flexible Pipeline for Complete Analysis of Bacterial Genomes.Segata N, editor. *mSystems* 5:e00190-20.

Pierce NT, Irber L, Reiter T, Brooks P, Brown CT. 2019. Large-scale sequence comparisons with sourmash. *F1000Res* 8:1006.

Qin Q-L, Xie B-B, Zhang X-Y, Chen X-L, Zhou B-C, Zhou J, Oren A, Zhang Y-Z. 2014. A Proposed Genus Boundary for the Prokaryotes Based on genomic insights. *J Bacteriol* 196:2210–2215.

Rangel-Pineros G, Almeida A, Beracochea M, Sakharova E, Marz M, Reyes Muñoz A, Hölzer M, Finn RD. 2023. VIRify: An integrated detection, annotation and taxonomic classification pipeline using virus-specific protein profile hidden Markov models.Ouzounis CA, editor. *PLoS Comput Biol* 19:e1011422.

Revell LJ. 2024. phytools 2.0: an updated R ecosystem for phylogenetic comparative methods (and other things). *PeerJ* 12:e16505.

Sánchez-Busó L, Yeats CA, Taylor B, Goater RJ, Underwood A, Abudahab K, Argimón S, Ma KC, Mortimer TD, Golparian D, et al. 2021. A community-driven resource for genomic epidemiology and antimicrobial resistance prediction of *Neisseria gonorrhoeae* at pathogenwatch. *Genome Med* 13:61.

Schliep KP. 2011. phangorn: phylogenetic analysis in R. *Bioinformatics* 27:592–593.

Tian L, Huang C, Mazloom R, Heath LS, Vinatzer BA. 2020. LINbase: a web server for genome-based identification of prokaryotes as members of crowdsourced taxa. *Nucleic Acids Research* 48:W529–W537.

Walker MA, Pedamallu CS, Ojesina AI, Bullman S, Sharpe T, Whelan CW, Meyerson M. 2018. GATK PathSeq: a customizable computational tool for the discovery and identification of microbial sequences in libraries from eukaryotic hosts.Berger B, editor. *Bioinformatics* 34:4287–4289.

Weisberg AJ, Grünwald NJ, Savory EA, Putnam ML, Chang JH. 2021. Genomic approaches to plant-pathogen epidemiology and diagnostics. *Annu. Rev. Phytopathol.* 59:311–332.

Williams DJ, Grimont PAD, Cazares A, Grimont F, Ageron E, Pettigrew KA, Cazares D, Njamkepo E, Weill F-X, Heinz E, et al. 2022. The genus *Serratia* revisited by genomics. *Nat Commun* 13:5195.

Yu G, Smith DK, Zhu H, Guan Y, Lam TT. 2017. ggtree: an R package for visualization and annotation of phylogenetic trees with their covariates and other associated data.McInerny G, editor. *Methods Ecol Evol* 8:28–36.

Figures

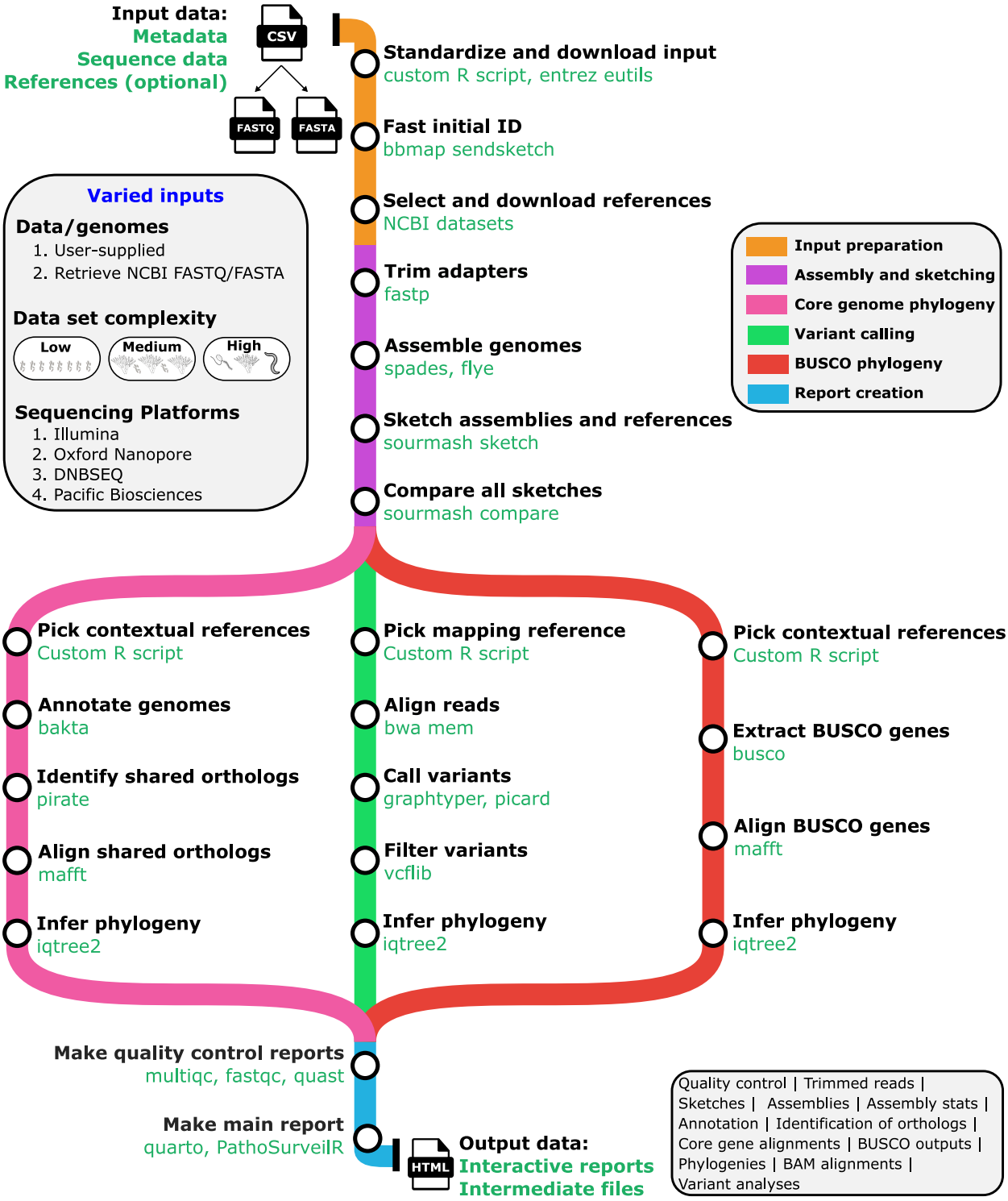

Figure 1. *PathogenSurveillance* automates comprehensive analysis of prokaryotic and eukaryotic whole genome sequencing data. A TSV file indicating paths or URLs to FASTQ or

FASTA files of short or long read sequencing data initiates the process of downloading and formatting of input files and references. The next part of the main branch quality controls sequencing reads and generates assemblies as well as *k*-mer sketches. It then diverges into three branches. The left branch is for samples of prokaryotic origin and will generate core phylogenies based on all shared orthologous gene sequences and variants, respectively, for each group of related samples. The right branch is for samples of eukaryotic origin and will build a phylogenetic tree for each group of related samples based on shared, single-copy BUSCO gene sequences. The middle branch creates SNP phylogenies for both prokaryotic and eukaryotic samples. Finally, the branches converge and *PathogenSurveillance* will generate a final HTML report with interactive plots and tables as well as quality control reports. Prominent steps in the workflow are printed in black beside each node with select corresponding tools used printed below in green.

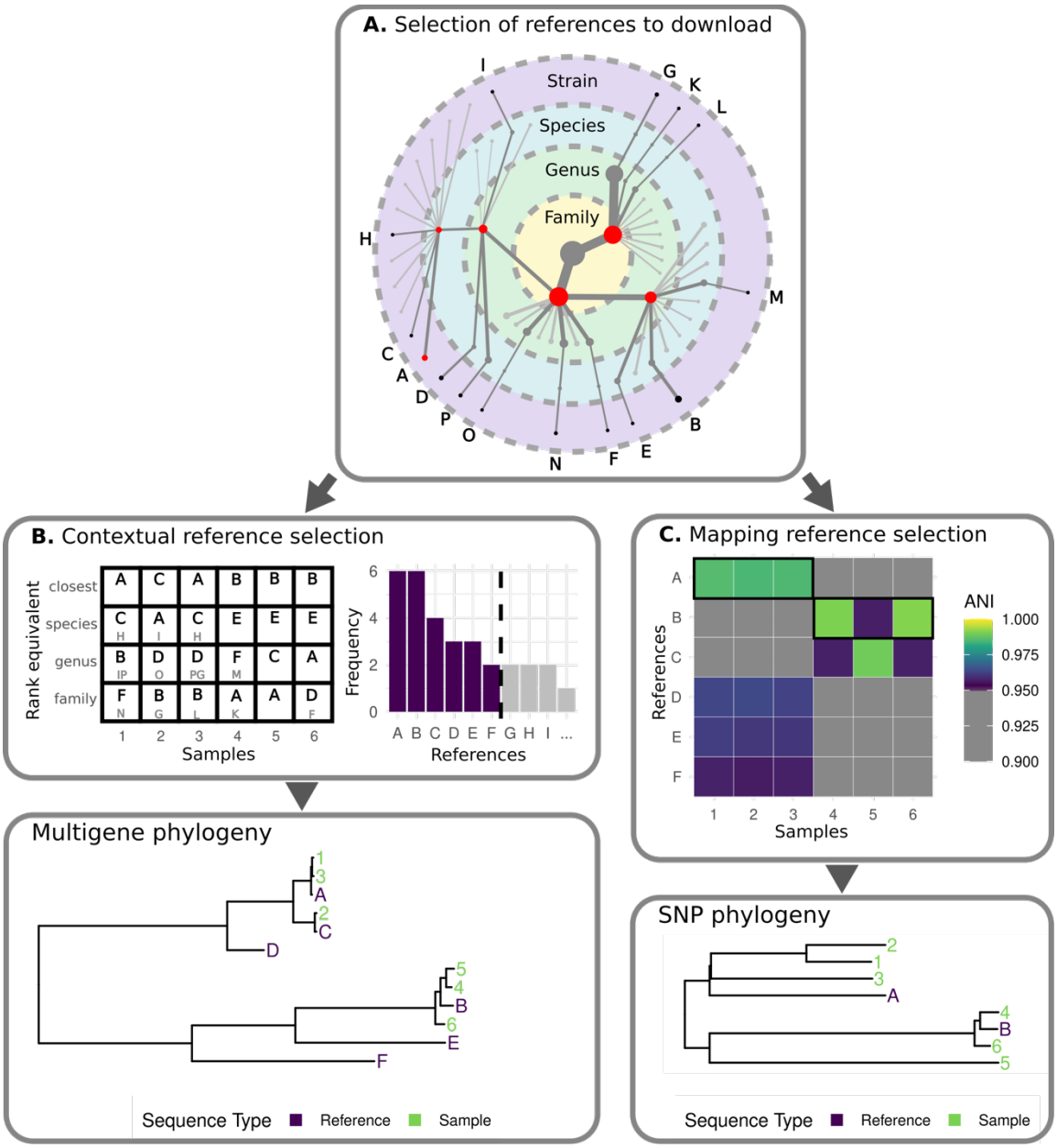

Figure 2. *PathogenSurveillance* automates reference selection for phylogenetic analyses. Shown is a hypothetical example illustrating sampling of genomes from **(A)** publicly available references for **(B)** multigene phylogenies and **(C)** SNP-based phylogenies or minimum spanning networks, respectively. **(A)** First, *PathogenSurveillance* compares *k*-mer sketches of raw reads to sketches of all NCBI RefSeq assemblies using BBmap sendsketch. Taxa likely present in samples are identified based on match similarity metrics (shown as red nodes in the taxonomic tree in (**A**) and representative assemblies of subtaxa are downloaded for each detected taxon (black nodes). The pipeline prioritizes enrichment of candidate references from more closely related taxa but also includes more distant taxa in case the initial identification was incorrect. **(B)** From the set of references downloaded a subset are selected to provide context for multi-gene phylogenetic trees based on pairwise ANI estimates between samples and references. References estimated to be most similar to each sample are always included in addition to a minimal set of

references selected for the set of samples as a whole that occur at a range of similarities to each sample. Each cell in the matrix represents a role that references can fit into. References are ordered by how many cells each can fill, as is shown in the histogram, and selected in this order until all cells are filled. **(C)** Mapping references are also selected for SNP calling based on estimated ANI values. A conceptual heatmap illustrates pairwise ANI comparisons, with cells exceeding the minimum threshold required shown in color. The reference with an estimated ANI exceeding a threshold for the most samples is chosen and this process is repeated until all samples have an assigned reference. Ties are broken based on mean ANI. A conceptual SNP-based phylogenetic tree demonstrates that only clusters with mapping references A and B met the minimum group size criteria for SNP calling and phylogenetic inference.

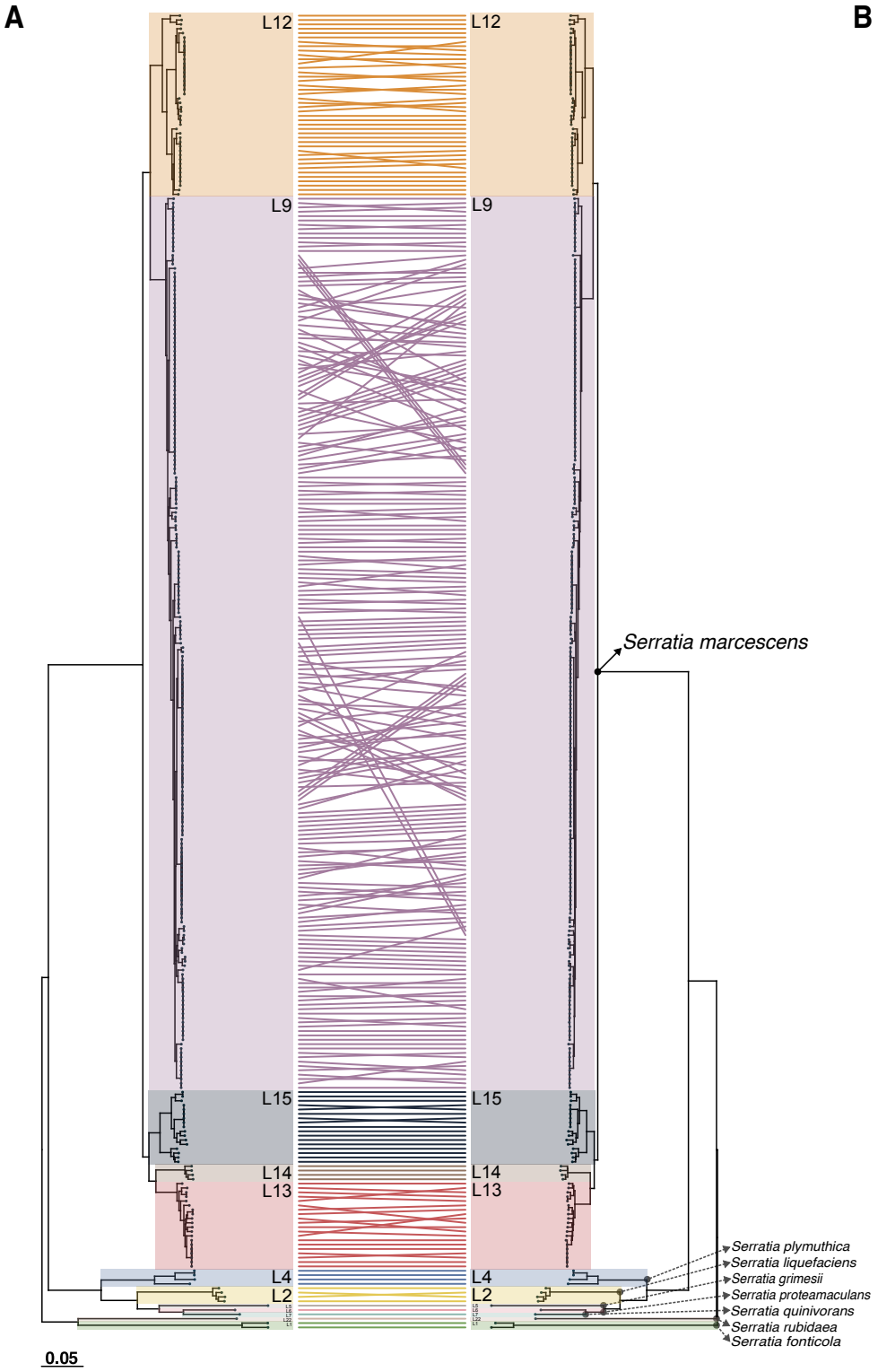

Figure 3. *PathogenSurveillance* core genome phylogeny reproduces a previously published phylogeny fully resolving species and lineages. The tanglegram shows congruence between (**A**) a maximum likelihood core gene phylogeny generated by *PathogenSurveillance*, and (**B**) a phylogeny reproduced from the literature. Matched samples are linked across trees by lines, with color-coding denoting lineage assignments. Major species and lineages (L#) are labeled as

published. Scale bars represent the number of nucleotide substitutions per site. See narrative for reference citation.

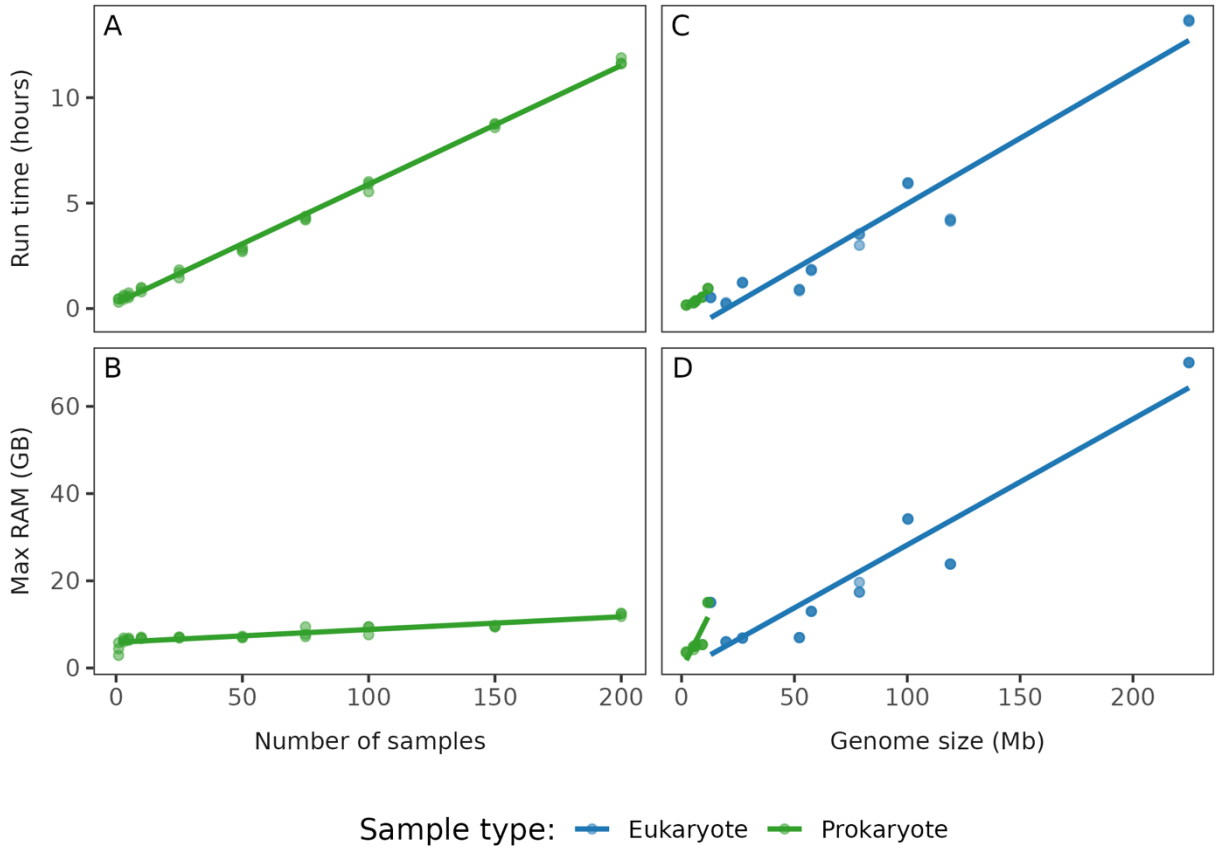

Figure 4. *PathogenSurveillance* demonstrates linear and scalable performance across a wide range of sample and genome sizes. Two test datasets were used to evaluate how the RAM and runtime of the pipeline scaled with number of samples and genome size. (**A**) and (**B**) show results for datasets increasing in sample sizes, while (**C**) and (**D**) show results for datasets with increasing genome sizes. Runtime (in clock hours) is plotted in (**A**) and (**C**), and maximum random-access memory (RAM) usage (in gigabytes (GB)) is plotted in (**B**) and (**D**). For genome size tests, each data set consisted of three samples of a given species and eukaryotes and prokaryotes are summarized separately since they are analyzed with different methods. Each dataset was analyzed in triplicate by *PathogenSurveillance*.

Supplementary data

```
output_directory
├── aligned_reads
│   ├── <reference_id>--<sample_id>.bam                                  : Unfiltered read alignements to references in BAM format
│   ├── <reference_id>--<sample_id>.formatted.bam                        : Quality filtered read alignements to references in BAM format
│   ├── <reference_id>--<sample_id>.formatted.bam.csi                    : Index for the filtered BAM files produced by samtools index
│   └── <reference_id>--<sample_id>.formatted.MarkDuplicates.metrics.txt : Output from picard MarkDuplicates containing summary data tables
├── annotations
│   └── <method>
│       ├── <sample_id>.embl                                             : Genome annotations and sequences in (multi) EMBL format
│       ├── <sample_id>.faa                                              : CDS/ORF amino acid sequences in FASTA format
│       ├── <sample_id>.ffn                                              : Feature nucleotide sequences in FASTA format
│       ├── <sample_id>.fna                                              : Replicon/contig DNA sequences in FASTA format
│       ├── <sample_id>.gbff                                             : Annotations of genomes in (multi) GenBank format
│       ├── <sample_id>.gff3                                             : Annotations of genomes in GFF3 format
│       ├── <sample_id>.hypotheticals.faa                                : Hypothetical protein CDS amino acid sequences in FASTA format
│       ├── <sample_id>.hypotheticals.tsv                                : Information on hypothetical protein CDSs in TSV format
│       ├── <sample_id>.tsv                                              : Information for protein CDSs in TSV format
│       └── <sample_id>.txt                                              : Genome annotation summary statistics in plain text format
├── assemblies
│   ├── flye
│   │   ├── <sample_id>.assembly_graph.gfa.gz                            : Genome assembly graphs in gzipped GFA format
│   │   ├── <sample_id>.assembly_graph.gv.gz                             : Genome assembly in gzipped Graphviz format
│   │   ├── <sample_id>.assembly_info.txt                                : Statistics on contigs produced by Flye in TSV fomrat
│   │   ├── <sample_id>.assembly.fasta.gz                                : Genome assemblies in gzipped FASTA format
│   │   ├── <sample_id>.flye.log                                         : Log file produced by Flye in plain text format
│   │   └── <sample_id>.params.json                                      : Parameters used to run Flye in JSON format
│   └── spades
│       ├── <sample_id>.assembly.gfa.gz                                  : Genome assembly graphs in gzipped GFA format
│       ├── <sample_id>.scaffolds.fa.gz                                  : Genome assemblies in gzipped FASTA format
│       ├── <sample_id>.spades.log                                       : Log file produced by Spades in plain text format
│       └── <sample_id>.warnings.log                                     : Warnings produced by Spades in plain text format
├── busco
│   ├── <sample_id>--<database>-busco                                    : BUSCO output directory
│   ├── <sample_id>--<database>-busco.batch_summary.txt                  : BUSCO analysis summary statistics in TSV format
│   └── <sample_id>--<database>-busco.log                                : Log file produced by BUSCO in plain text format
├── downloads
│   ├── annotations
│   │   └── <reference_id>.gff.gz                                        : Genome annotations downloaded by the pipeline in gzipped GFF format
│   ├── assemblies
│   │   └── <reference_id>.fasta.gz                                      : Genome assemblies downloaded by the pipeline in gzipped FASTA format
│   └── reads
│       └── <reference_id>_<orientation>.fastq.gz                        : Reads downloaded by the pipeline in gzipped FASTQ format
├── fastp
│   ├── <sample_id>.fastp.html                                           : FASTP report with read quality and adapter trimming statistics in HTML format
│   ├── <sample_id>.fastp.json                                           : FASTP report with read quality and adapter trimming statistic in JSON format
│   └── <sample_id>.fastp.log                                            : Log file produced by FASTP
├── metadata
│   ├── reference_metadata.tsv                                           : Reference metadata in TSV format
│   └── sample_metadata.tsv                                              : Sample metadata in TSV format
├── pipeline_info
│   ├── execution_report_<date>.html                                     : Statistics like time and ram usage for pipeline execution as a HTML report
│   ├── execution_trace_<date>.txt                                       : Statistics like time and ram usage for pipeline execution in TSV format
│   ├── messages.tsv                                                     : Errors, warnings, and messages produced by the pipeline in TSV format
│   ├── params_<date>.json                                               : Parameters used when running the pipeline in JSON format
│   ├── pathogensurveillance_run_info.yml                                : Pipeline run metadata in YML format
│   ├── pipeline_dag_<date>.html                                         : Flowchart of the pathogensurveillance pipeline automatically generated by Nextflow
│   └── version_info.yml                                                 : Versions of software used in the pipeline in YML format
├── pirate
│   └── <report_id>_results                                              : Pirate output directory with gene orthology information
├── pocp
│   └── <report_id>_pocp.tsv                                             : Percentage of conserved proteins in TSV format
├── quality_control
│   ├── fastqc
│   │   ├── <reads_id>_fastqc.html                                       : FastQC report on read quality in HTML format
│   │   └── <reads_id>_fastqc.zip                                        : Zipped directory of FastQC output
│   ├── multiqc
│   │   └── <report_id>_multiqc                                          : Directory of MultiQC output with intermediate files used to make the MultiQC report
│   │       └── multiqc_report.html                                      : MultiQC report combining the summary quality control output in HTML format
│   ├── nanoplot
│   │   ├── <sample_id>_nanoplot-report.html                             : Nanoplot report on read quality in HTML format
│   │   └── <sample_id>_NanoStats.txt                                    : Nanoplot report on read quality in plain text format
│   └── quast
│       ├── <sample_id>.tsv                                              : Quast summary statistics on genome assembly quality in headerless TSV format
│       └── <sample_id>
│           └── report.html                                              : Quast report with data on genome assembly quality in HTML format
├── reference_data
│   ├── considered
│   │   └── <taxon_id>.tsv                                               : Metadata of available assemblies on NCBI in TSV format
│   ├── downloaded
│   │   └── <sample_id>.tsv                                              : Metadata of references selected for download from NCBI in TSV format
│   └── selected
│       ├── <report_id>_busco_references.tsv                             : IDs of references selected for multigene phylogenies in TSV format
│       ├── <report_id>_core_references.tsv                              : IDs of references selected for multigene phylogenies in TSV format
│       └── <report_id>_mapping_references.tsv                           : IDs of references selected for multigene phylogenies in TSV format
├── reports
│   └── <report_id>_pathsurveil_report.html                              : Primary report for pathogensurveillance in HTML format
├── sendsketch
│   └── <sample_id>.txt                                                  : Table returned by BBmap sendsketch with initial identifications in TSV format
├── shared_genes
│   └── <method>
│       └── <gene_id>.aligned.fas                                        : Aligned sequences for each gene shared amoung a group of organisms in FASTA format
├── sketch_comparisons
│   ├── ani_matricies
│   │   └── <report_id>_comp.csv                                         : Estimated ANI similarity matrix made by sourmash compare in CSV format
│   └── sketches
│       └── <id>.sig                                                     : FracMinHash signature of the given sequence made by sourmash sketch in JSON
├── trees
│   ├── busco
│   │   └── <report_id>--<cluster_id>.treefile                           : Trees infered from multigene phylogeny in Newick format
│   ├── core
│   │   └── <report_id>--<cluster_id>.treefile                           : Trees infered from multigene phylogeny in Newick format
│   └── snp
│       └── <report_id>--<reference_id>.treefile                         : Trees inferred from variant analysis in Newick format
└── variants
    ├── <report_id>--<reference_id>.fasta                                : Filtered variants in FASTA format
    ├── <report_id>--<reference_id>.vcf.gz                               : Unfiltered variants in VCF format
    └── <report_id>--<reference_id>.vcffilter.vcf.gz                     : Filtered variants in VCF format
```

Supplementary Figure 1. *PathogenSurveillance* organizes and names output files for further analyses. The left side displays the hierarchical structure of output files. Angle brackets (<>) show run specific identifiers for references, samples, and reports. The right side provides brief descriptions of each file type, highlighting their applications in downstream analyses.

Supplementary Table 1. Modules and module updates contributed to nf-core as part of the PathogenSurveillance pipeline development.

Nf-core module	**Status**	**Actions**
`bbmap/sendsketch`	New	Added new module
`busco/download`	New	Added new module
`graphtyper/genotype`	New	Added new module
`graphtyper/vcf_concatenate`	New	Added new module
`khmer/trimlowabund`	New	Added new module
`seqkit/head`	New	Added new module
`vcflib/vcffilter`	New	Added new module
`busco/busco`	Update	Added single copy sequence files to the usable output files
`gatk4/variantfiltration`	Update	Allowed for the use of compressed reference files
`iqtree`	Update	Allowed for the use of multiple input alignments
`pirate`	Update	Allowed for the use of GFF3 files and updated dependencies
`sourmash/compare`	Update	Modified script to name output by prefix and updated dependencies
`sourmash/gather`	Update	Created new unit tests
`sourmash/taxannotate`	Update	Created new unit tests
`tabix/bgzip`	Update	Made index file parameter detection more robust

Supplementary Table 2. Accessions of the *Serratia* genus with both long-read Nanopore and short-read Illumina data used for validation of the *PathogenSurveillance* pipeline from (Williams et al. 2022).

Sample	NCBI accession	BioProject	Lineage	Phylogroup	Sequence type
1	SRR3927533	PRJNA60453	4	8	Illumina
2	SRR3927534	PRJNA60457	4	8	Illumina
3	SRR3926812	PRJNA178957	22	11	Illumina
4	SRR3927990	PRJNA61833	7	1	Illumina
5	SRR3925616	PRJNA175925	4	8	Illumina
6	SRR5629779	PRJNA246471	12	2	Nanopore
7	SRR2059988	PRJNA286707	13	2	Illumina
8	SRR1655333	PRJNA267549	12	2	Illumina
9	SRR1655414	PRJNA267549	9	2	Illumina
10	SRR1655439	PRJNA267549	13	2	Illumina
11	SRR1655440	PRJNA267549	13	2	Illumina
12	SRR1655376	PRJNA267549	15	2	Illumina
13	SRR1655391	PRJNA267549	9	2	Illumina
14	SRR1655415	PRJNA267549	9	2	Illumina
15	SRR1655375	PRJNA267549	15	2	Illumina
16	SRR1655390	PRJNA267549	9	2	Illumina
17	SRR1655624	PRJNA267549	2	4	Illumina
18	SRR1655701	PRJNA267549	15	2	Illumina
19	SRR1655706	PRJNA267549	15	2	Illumina
20	SRR1655712	PRJNA267549	15	2	Illumina
21	SRR1655833	PRJNA267549	13	2	Illumina
22	SRR1655844	PRJNA267549	13	2	Illumina
23	SRR1655713	PRJNA267549	15	2	Illumina
24	SRR1655716	PRJNA267549	15	2	Illumina
25	SRR1655733	PRJNA267549	15	2	Illumina
26	SRR1655793	PRJNA267549	13	2	Illumina
27	SRR1655797	PRJNA267549	13	2	Illumina
28	SRR1655818	PRJNA267549	13	2	Illumina
29	SRR1655820	PRJNA267549	13	2	Illumina
30	SRR1655878	PRJNA267549	13	2	Illumina
31	SRR1655960	PRJNA267549	15	2	Illumina
32	SRR1655976	PRJNA267549	15	2	Illumina
33	SRR1656119	PRJNA267549	12	2	Illumina
34	SRR1656133	PRJNA267549	9	2	Illumina
35	SRR1656143	PRJNA267549	9	2	Illumina

36	SRR1656234	PRJNA267549	9	2	Illumina
37	SRR1656346	PRJNA267549	12	2	Illumina
38	ERR486640	PRJEB5065	9	2	Illumina
39	ERR486562	PRJEB5065	9	2	Illumina
40	ERR486664	PRJEB5065	9	2	Illumina
41	ERR486618	PRJEB5065	9	2	Illumina
42	ERR486619	PRJEB5065	9	2	Illumina
43	ERR486680	PRJEB5065	9	2	Illumina
44	ERR486558	PRJEB5065	9	2	Illumina
45	ERR486565	PRJEB5065	9	2	Illumina
46	ERR486521	PRJEB5065	9	2	Illumina
47	ERR486559	PRJEB5065	9	2	Illumina
48	ERR486637	PRJEB5065	9	2	Illumina
49	ERR486676	PRJEB5065	9	2	Illumina
50	ERR486823	PRJEB5065	9	2	Illumina
51	ERR486814	PRJEB5065	9	2	Illumina
52	ERR486490	PRJEB5065	12	2	Illumina
53	ERR486644	PRJEB5065	9	2	Illumina
54	ERR486661	PRJEB5065	9	2	Illumina
55	ERR486645	PRJEB5065	9	2	Illumina
56	ERR486581	PRJEB5065	9	2	Illumina
57	ERR486649	PRJEB5065	9	2	Illumina
58	ERR486604	PRJEB5065	9	2	Illumina
59	ERR486674	PRJEB5065	9	2	Illumina
60	ERR486561	PRJEB5065	9	2	Illumina
61	ERR486620	PRJEB5065	9	2	Illumina
62	ERR486635	PRJEB5065	9	2	Illumina
63	ERR486651	PRJEB5065	9	2	Illumina
64	ERR486631	PRJEB5065	9	2	Illumina
65	ERR486574	PRJEB5065	9	2	Illumina
66	ERR486636	PRJEB5065	9	2	Illumina
67	ERR486549	PRJEB5065	9	2	Illumina
68	ERR486812	PRJEB5065	9	2	Illumina
69	ERR486603	PRJEB5065	9	2	Illumina
70	ERR486681	PRJEB5065	13	2	Illumina
71	ERR486488	PRJEB5065	9	2	Illumina
72	ERR486610	PRJEB5065	9	2	Illumina
73	ERR486525	PRJEB5065	9	2	Illumina
74	ERR486818	PRJEB5065	9	2	Illumina
75	ERR486616	PRJEB5065	9	2	Illumina

76	ERR486533	PRJEB5065	12	2	Illumina
77	ERR486567	PRJEB5065	9	2	Illumina
78	ERR486572	PRJEB5065	12	2	Illumina
79	ERR486594	PRJEB5065	9	2	Illumina
80	ERR486487	PRJEB5065	9	2	Illumina
81	ERR486816	PRJEB5065	9	2	Illumina
82	ERR486548	PRJEB5065	9	2	Illumina
83	ERR486573	PRJEB5065	9	2	Illumina
84	ERR486596	PRJEB5065	9	2	Illumina
85	ERR486593	PRJEB5065	12	2	Illumina
86	ERR486497	PRJEB5065	9	2	Illumina
87	ERR486543	PRJEB5065	9	2	Illumina
88	ERR486480	PRJEB5065	9	2	Illumina
89	ERR486577	PRJEB5065	9	2	Illumina
90	ERR486570	PRJEB5065	9	2	Illumina
91	ERR486810	PRJEB5065	9	2	Illumina
92	ERR486522	PRJEB5065	9	2	Illumina
93	ERR486608	PRJEB5065	9	2	Illumina
94	ERR486496	PRJEB5065	9	2	Illumina
95	ERR486659	PRJEB5065	9	2	Illumina
96	ERR486554	PRJEB5065	9	2	Illumina
97	ERR486617	PRJEB5065	9	2	Illumina
98	ERR486535	PRJEB5065	9	2	Illumina
99	ERR486580	PRJEB5065	9	2	Illumina
100	ERR486807	PRJEB5065	9	2	Illumina
101	ERR486670	PRJEB5065	9	2	Illumina
102	ERR486598	PRJEB5065	12	2	Illumina
103	ERR486683	PRJEB5065	12	2	Illumina
104	ERR486526	PRJEB5065	9	2	Illumina
105	ERR486473	PRJEB5065	9	2	Illumina
106	ERR486825	PRJEB5065	9	2	Illumina
107	ERR486492	PRJEB5065	9	2	Illumina
108	ERR486528	PRJEB5065	9	2	Illumina
109	ERR486820	PRJEB5065	9	2	Illumina
110	ERR486530	PRJEB5065	9	2	Illumina
111	ERR486809	PRJEB5065	9	2	Illumina
112	ERR486677	PRJEB5065	9	2	Illumina
113	ERR486634	PRJEB5065	9	2	Illumina
114	ERR486656	PRJEB5065	9	2	Illumina
115	ERR486485	PRJEB5065	9	2	Illumina

116	ERR486811	PRJEB5065	9	2	Illumina
117	ERR486817	PRJEB5065	9	2	Illumina
118	ERR486587	PRJEB5065	9	2	Illumina
119	ERR486586	PRJEB5065	9	2	Illumina
120	ERR486806	PRJEB5065	9	2	Illumina
121	ERR486653	PRJEB5065	9	2	Illumina
122	ERR486569	PRJEB5065	9	2	Illumina
123	ERR486551	PRJEB5065	9	2	Illumina
124	ERR486477	PRJEB5065	9	2	Illumina
125	ERR486486	PRJEB5065	12	2	Illumina
126	ERR486633	PRJEB5065	9	2	Illumina
127	ERR486529	PRJEB5065	9	2	Illumina
128	ERR486482	PRJEB5065	9	2	Illumina
129	ERR486684	PRJEB5065	9	2	Illumina
130	ERR486663	PRJEB5065	9	2	Illumina
131	ERR486527	PRJEB5065	9	2	Illumina
132	ERR486539	PRJEB5065	9	2	Illumina
133	ERR486605	PRJEB5065	9	2	Illumina
134	ERR486821	PRJEB5065	9	2	Illumina
135	ERR486652	PRJEB5065	9	2	Illumina
136	ERR486494	PRJEB5065	9	2	Illumina
137	ERR486555	PRJEB5065	9	2	Illumina
138	ERR486531	PRJEB5065	9	2	Illumina
139	ERR486489	PRJEB5065	12	2	Illumina
140	ERR486550	PRJEB5065	9	2	Illumina
141	ERR486582	PRJEB5065	9	2	Illumina
142	ERR486678	PRJEB5065	9	2	Illumina
143	ERR486675	PRJEB5065	9	2	Illumina
144	ERR486621	PRJEB5065	9	2	Illumina
145	ERR486520	PRJEB5065	9	2	Illumina
146	ERR486599	PRJEB5065	9	2	Illumina
147	ERR486623	PRJEB5065	9	2	Illumina
148	ERR486628	PRJEB5065	9	2	Illumina
149	ERR486498	PRJEB5065	9	2	Illumina
150	ERR486808	PRJEB5065	12	2	Illumina
151	ERR486524	PRJEB5065	9	2	Illumina
152	ERR486625	PRJEB5065	12	2	Illumina
153	ERR486822	PRJEB5065	12	2	Illumina
154	ERR486602	PRJEB5065	9	2	Illumina
155	ERR486481	PRJEB5065	9	2	Illumina

156	ERR486478	PRJEB5065	9	2	Illumina
157	ERR486606	PRJEB5065	9	2	Illumina
158	ERR486476	PRJEB5065	9	2	Illumina
159	ERR486647	PRJEB5065	9	2	Illumina
160	ERR486479	PRJEB5065	9	2	Illumina
161	ERR486614	PRJEB5065	9	2	Illumina
162	ERR486556	PRJEB5065	9	2	Illumina
163	ERR486626	PRJEB5065	9	2	Illumina
164	ERR486474	PRJEB5065	12	2	Illumina
165	ERR486641	PRJEB5065	9	2	Illumina
166	ERR486583	PRJEB5065	9	2	Illumina
167	ERR486553	PRJEB5065	9	2	Illumina
168	ERR486609	PRJEB5065	9	2	Illumina
169	ERR486627	PRJEB5065	9	2	Illumina
170	ERR486576	PRJEB5065	9	2	Illumina
171	ERR486650	PRJEB5065	13	2	Illumina
172	ERR486824	PRJEB5065	9	2	Illumina
173	ERR486475	PRJEB5065	9	2	Illumina
174	ERR486536	PRJEB5065	9	2	Illumina
175	ERR486826	PRJEB5065	9	2	Illumina
176	ERR486638	PRJEB5065	9	2	Illumina
177	ERR486592	PRJEB5065	9	2	Illumina
178	ERR486564	PRJEB5065	9	2	Illumina
179	ERR486668	PRJEB5065	9	2	Illumina
180	ERR486632	PRJEB5065	9	2	Illumina
181	ERR486541	PRJEB5065	9	2	Illumina
182	ERR486639	PRJEB5065	9	2	Illumina
183	ERR486667	PRJEB5065	9	2	Illumina
184	ERR486546	PRJEB5065	9	2	Illumina
185	ERR486542	PRJEB5065	9	2	Illumina
186	ERR486588	PRJEB5065	9	2	Illumina
187	ERR486560	PRJEB5065	9	2	Illumina
188	ERR486532	PRJEB5065	9	2	Illumina
189	ERR486538	PRJEB5065	9	2	Illumina
190	ERR486662	PRJEB5065	12	2	Illumina
191	ERR486568	PRJEB5065	9	2	Illumina
192	ERR486601	PRJEB5065	9	2	Illumina
193	ERR486566	PRJEB5065	9	2	Illumina
194	ERR486646	PRJEB5065	9	2	Illumina
195	ERR486575	PRJEB5065	9	2	Illumina

196	ERR486648	PRJEB5065	9	2	Illumina
197	ERR486534	PRJEB5065	9	2	Illumina
198	ERR486495	PRJEB5065	15	2	Illumina
199	ERR486643	PRJEB5065	9	2	Illumina
200	ERR486629	PRJEB5065	12	2	Illumina
201	ERR486665	PRJEB5065	9	2	Illumina
202	ERR486672	PRJEB5065	9	2	Illumina
203	ERR486815	PRJEB5065	9	2	Illumina
204	ERR486600	PRJEB5065	9	2	Illumina
205	ERR486499	PRJEB5065	9	2	Illumina
206	ERR486622	PRJEB5065	9	2	Illumina
207	ERR486679	PRJEB5065	9	2	Illumina
208	ERR486537	PRJEB5065	9	2	Illumina
209	ERR486545	PRJEB5065	12	2	Illumina
210	ERR486493	PRJEB5065	9	2	Illumina
211	ERR486612	PRJEB5065	9	2	Illumina
212	ERR486552	PRJEB5065	9	2	Illumina
213	ERR486611	PRJEB5065	9	2	Illumina
214	ERR486589	PRJEB5065	9	2	Illumina
215	ERR486590	PRJEB5065	9	2	Illumina
216	ERR486655	PRJEB5065	9	2	Illumina
217	ERR486563	PRJEB5065	9	2	Illumina
218	ERR486579	PRJEB5065	9	2	Illumina
219	ERR486819	PRJEB5065	9	2	Illumina
220	ERR486557	PRJEB5065	9	2	Illumina
221	ERR486660	PRJEB5065	9	2	Illumina
222	ERR486813	PRJEB5065	12	2	Illumina
223	ERR486613	PRJEB5065	9	2	Illumina
224	ERR486666	PRJEB5065	9	2	Illumina
225	ERR486571	PRJEB5065	9	2	Illumina
226	ERR486654	PRJEB5065	9	2	Illumina
227	ERR486682	PRJEB5065	9	2	Illumina
228	ERR486673	PRJEB5065	9	2	Illumina
229	ERR486578	PRJEB5065	9	2	Illumina
230	ERR486584	PRJEB5065	9	2	Illumina
231	ERR486484	PRJEB5065	9	2	Illumina
232	ERR486483	PRJEB5065	9	2	Illumina
233	ERR486597	PRJEB5065	9	2	Illumina
234	ERR486658	PRJEB5065	9	2	Illumina
235	ERR486547	PRJEB5065	9	2	Illumina

236	ERR486595	PRJEB5065	9	2	Illumina
237	ERR486591	PRJEB5065	9	2	Illumina
238	ERR486671	PRJEB5065	9	2	Illumina
239	ERR486540	PRJEB5065	9	2	Illumina
240	ERR486642	PRJEB5065	9	2	Illumina
241	ERR486669	PRJEB5065	9	2	Illumina
242	SRR3439014	PRJNA319063	1	3	Illumina
243	SRR4289877	PRJNA338688	1	3	Illumina
244	SRR1707029	PRJNA270035	12	2	Illumina
245	SRR5330012	PRJNA348753	15	2	Illumina
246	SRR5902281	PRJNA396838	9	2	Illumina
247	SRR5902258	PRJNA396838	9	2	Illumina
248	SRR5902283	PRJNA396838	14	2	Illumina
249	SRR5902279	PRJNA396838	9	2	Illumina
250	SRR5902263	PRJNA396838	9	2	Illumina
251	SRR6676029	PRJNA430813	12	2	Illumina
252	SRR6676031	PRJNA430813	12	2	Illumina
253	SRR6675995	PRJNA430813	12	2	Illumina
254	SRR7160685	PRJNA411762	12	2	Illumina
255	SRR6487756	PRJNA417962	2	4	Illumina
256	SRR6488026	PRJNA417962	5	7	Illumina
257	SRR6482683	PRJNA417962	6	9	Illumina
258	SRR6482683	PRJNA417962	6	9	Illumina
259	SRR5415137	PRJNA381635	15	2	Illumina
260	SRR5415011	PRJNA381609	14	2	Illumina
261	ERR1711267	PRJEB16676	12	2	Illumina
262	ERR9821170	PRJEB51113	9	2	Illumina
263	ERR9821139	PRJEB51113	14	2	Illumina
264	ERR9821145	PRJEB51113	9	2	Illumina
265	ERR9821130	PRJEB51113	15	2	Illumina
266	ERR9821158	PRJEB51113	15	2	Illumina
267	ERR9821156	PRJEB51113	13	2	Illumina
268	ERR9821162	PRJEB51113	15	2	Illumina
269	ERR9821140	PRJEB51113	13	2	Illumina
270	ERR9821141	PRJEB51113	15	2	Illumina
271	ERR9821159	PRJEB51113	9	2	Illumina
272	ERR9821169	PRJEB51113	9	2	Illumina
273	ERR9821161	PRJEB51113	9	2	Illumina
274	ERR9821136	PRJEB51113	12	2	Illumina
275	ERR9821142	PRJEB51113	12	2	Illumina

276	ERR9821153	PRJEB51113	12	2	Illumina
277	ERR9821135	PRJEB51113	12	2	Illumina
278	ERR9821138	PRJEB51113	12	2	Illumina
279	ERR9821157	PRJEB51113	12	2	Illumina
280	ERR9821151	PRJEB51113	9	2	Illumina
281	ERR9821155	PRJEB51113	12	2	Illumina
282	ERR9821131	PRJEB51113	12	2	Illumina
283	ERR9821143	PRJEB51113	12	2	Illumina
284	ERR9821165	PRJEB51113	9	2	Illumina
285	ERR9821164	PRJEB51113	12	2	Illumina
286	ERR9821147	PRJEB51113	13	2	Illumina
287	ERR9821166	PRJEB51113	12	2	Illumina
288	ERR9821168	PRJEB51113	9	2	Illumina
289	ERR9821146	PRJEB51113	12	2	Illumina
290	ERR9821149	PRJEB51113	9	2	Illumina
291	ERR9821148	PRJEB51113	14	2	Illumina
292	ERR9821167	PRJEB51113	12	2	Illumina
293	ERR9821163	PRJEB51113	12	2	Illumina
294	ERR9821133	PRJEB51113	2	4	Illumina
295	ERR9821144	PRJEB51113	2	4	Illumina
296	ERR9821152	PRJEB51113	13	2	Illumina
297	ERR9821132	PRJEB51113	12	2	Illumina
298	ERR9821134	PRJEB51113	13	2	Illumina
299	ERR9821160	PRJEB51113	13	2	Illumina
300	ERR9821154	PRJEB51113	13	2	Illumina
301	ERR9821150	PRJEB51113	12	2	Illumina
302	ERR9821137	PRJEB51113	13	2	Illumina

Supplementary Table 4. Genbank accessions of *Klebsiella pneumoniae*. The associated Illumina reads were processed by the *PathogenSurveillance* pipeline for validation used in Figure 4A, B.

Sample_ID	NCBI accession	Bioproject	Species	Kingdom	Sequence type
ERR024818	ERR024818	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024819	ERR024819	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024821	ERR024821	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024822	ERR024822	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024823	ERR024823	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024824	ERR024824	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024828	ERR024828	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024831	ERR024831	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024832	ERR024832	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024833	ERR024833	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024835	ERR024835	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024836	ERR024836	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024837	ERR024837	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024838	ERR024838	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024839	ERR024839	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024840	ERR024840	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024843	ERR024843	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024844	ERR024844	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024845	ERR024845	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024847	ERR024847	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024848	ERR024848	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024850	ERR024850	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024851	ERR024851	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024852	ERR024852	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR024854	ERR024854	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025098	ERR025098	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025099	ERR025099	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025100	ERR025100	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025101	ERR025101	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025103	ERR025103	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025104	ERR025104	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025105	ERR025105	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025107	ERR025107	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025108	ERR025108	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025109	ERR025109	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina

ERR025111	ERR025111	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025112	ERR025112	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025113	ERR025113	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025115	ERR025115	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025116	ERR025116	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025117	ERR025117	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025118	ERR025118	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025119	ERR025119	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025121	ERR025121	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025125	ERR025125	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025128	ERR025128	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025131	ERR025131	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025132	ERR025132	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025135	ERR025135	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025137	ERR025137	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025139	ERR025139	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025140	ERR025140	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025141	ERR025141	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025142	ERR025142	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025143	ERR025143	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025144	ERR025144	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025146	ERR025146	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025147	ERR025147	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025150	ERR025150	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025151	ERR025151	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025152	ERR025152	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025154	ERR025154	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025155	ERR025155	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025156	ERR025156	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025157	ERR025157	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025158	ERR025158	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025159	ERR025159	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025160	ERR025160	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025161	ERR025161	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025462	ERR025462	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025464	ERR025464	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025465	ERR025465	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025468	ERR025468	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025469	ERR025469	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025470	ERR025470	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina

ERR025471	ERR025471	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025472	ERR025472	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025473	ERR025473	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025475	ERR025475	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025477	ERR025477	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025478	ERR025478	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025479	ERR025479	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025482	ERR025482	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025483	ERR025483	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025484	ERR025484	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025485	ERR025485	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025486	ERR025486	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025489	ERR025489	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025491	ERR025491	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025494	ERR025494	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025495	ERR025495	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025496	ERR025496	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025497	ERR025497	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025498	ERR025498	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025499	ERR025499	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025501	ERR025501	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025502	ERR025502	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025504	ERR025504	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025505	ERR025505	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025510	ERR025510	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025511	ERR025511	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025512	ERR025512	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025515	ERR025515	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025516	ERR025516	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025517	ERR025517	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025518	ERR025518	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025519	ERR025519	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025520	ERR025520	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025521	ERR025521	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025522	ERR025522	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025523	ERR025523	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025524	ERR025524	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025525	ERR025525	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025527	ERR025527	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025529	ERR025529	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina

ERR025531	ERR025531	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025532	ERR025532	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025534	ERR025534	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025535	ERR025535	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025536	ERR025536	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025538	ERR025538	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025540	ERR025540	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025541	ERR025541	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025542	ERR025542	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025543	ERR025543	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025544	ERR025544	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025545	ERR025545	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025546	ERR025546	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025547	ERR025547	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025548	ERR025548	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025550	ERR025550	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025553	ERR025553	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025554	ERR025554	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025555	ERR025555	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025557	ERR025557	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025558	ERR025558	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025561	ERR025561	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025562	ERR025562	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025564	ERR025564	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025566	ERR025566	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025570	ERR025570	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025571	ERR025571	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025572	ERR025572	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025574	ERR025574	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025575	ERR025575	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025576	ERR025576	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025579	ERR025579	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025581	ERR025581	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025582	ERR025582	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025583	ERR025583	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025585	ERR025585	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025587	ERR025587	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025588	ERR025588	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025589	ERR025589	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025590	ERR025590	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina

ERR025593	ERR025593	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025594	ERR025594	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025595	ERR025595	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025596	ERR025596	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025597	ERR025597	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025599	ERR025599	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025600	ERR025600	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025601	ERR025601	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025602	ERR025602	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025605	ERR025605	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025607	ERR025607	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025608	ERR025608	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025609	ERR025609	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025610	ERR025610	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025612	ERR025612	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025613	ERR025613	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025614	ERR025614	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025618	ERR025618	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025620	ERR025620	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025621	ERR025621	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025624	ERR025624	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025625	ERR025625	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025627	ERR025627	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025628	ERR025628	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025629	ERR025629	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025633	ERR025633	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025634	ERR025634	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025635	ERR025635	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025637	ERR025637	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025638	ERR025638	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025639	ERR025639	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025641	ERR025641	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025644	ERR025644	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025645	ERR025645	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025646	ERR025646	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025647	ERR025647	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025648	ERR025648	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025649	ERR025649	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025651	ERR025651	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025652	ERR025652	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina

ERR025653	ERR025653	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025654	ERR025654	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025655	ERR025655	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025657	ERR025657	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025658	ERR025658	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025659	ERR025659	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025660	ERR025660	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025661	ERR025661	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025662	ERR025662	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025663	ERR025663	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025664	ERR025664	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025666	ERR025666	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025667	ERR025667	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025668	ERR025668	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025673	ERR025673	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025674	ERR025674	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025675	ERR025675	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025676	ERR025676	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025677	ERR025677	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025678	ERR025678	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025679	ERR025679	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025680	ERR025680	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025979	ERR025979	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025980	ERR025980	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025984	ERR025984	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025985	ERR025985	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025988	ERR025988	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025989	ERR025989	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025992	ERR025992	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025993	ERR025993	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025994	ERR025994	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025995	ERR025995	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025996	ERR025996	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025998	ERR025998	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR025999	ERR025999	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR026000	ERR026000	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina
ERR026001	ERR026001	PRJEB2111	*Klebsiella pneumoniae*	Bacteria	Illumina

Supplementary Table 5. Genbank accessions of samples belonging to both prokaryotes and eukaryotes. The associated reads were processed by the *PathogenSurveillance* pipeline for validation used in Figure 4C, D.

Sample_ID	NCBI SRA accession	Bioproject	Species	Kingdom	Sequence type
Celg1	SRR28479892	PRJNA1091400	*Caenorhabditis elegans*	Anamalia	Illumina
Celg2	SRR28479941	PRJNA1091400	*Caenorhabditis elegans*	Anamalia	Illumina
Celg3	SRR28479942	PRJNA1091400	*Caenorhabditis elegans*	Anamalia	Illumina
Brady1	SRR17382571	PRJNA793169	*Bradyrhizobium japonicum*	Bacteria	Illumina
Brady2	SRR17382844	PRJNA793169	*Bradyrhizobium japonicum*	Bacteria	Illumina
Brady3	SRR17388438	PRJNA793169	*Bradyrhizobium japonicum*	Bacteria	Illumina
Limo1	ERR12342572	PRJEB70875	*Limosilactobacillus fermentum*	Bacteria	Illumina
Limo2	SRR18850315	PRJNA830347	*Limosilactobacillus fermentum*	Bacteria	Illumina
Limo3	SRR18850317	PRJNA830347	*Limosilactobacillus fermentum*	Bacteria	Illumina
Pseudo1	SRR33648670	PRJNA1225923	*Pseudomonas aeruginosa*	Bacteria	Illumina
Pseudo2	SRR32466794	PRJNA1225923	*Pseudomonas aeruginosa*	Bacteria	Illumina
Pseudo3	SRR33648579	PRJNA1225923	*Pseudomonas aeruginosa*	Bacteria	Illumina
Solivo1	SRR9661589	PRJNA553182	*Streptomyces olivochromogenes*	Bacteria	Illumina
Solivo2	SRR9661590	PRJNA553182	*Streptomyces olivochromogenes*	Bacteria	Illumina
Solivo3	SRR9661593	PRJNA553182	*Streptomyces olivochromogenes*	Bacteria	Illumina
Xan1	SRR28281444	PRJNA991976	*Xanthomonas hortorum pv pelargonii*	Bacteria	Illumina
Xan2	SRR28281445	PRJNA991976	*Xanthomonas hortorum pv pelargonii*	Bacteria	Illumina
Xan3	SRR28281439	PRJNA991976	*Xanthomonas hortorum pv pelargonii*	Bacteria	Illumina
Fus1	SRR21509025	PRJNA878477	*Fusarium oxysporum f sp niveum*	Fungi	Illumina
Fus2	SRR21509023	PRJNA878477	*Fusarium oxysporum f sp niveum*	Fungi	Illumina
Fus3	SRR21509022	PRJNA878477	*Fusarium oxysporum f sp niveum*	Fungi	Illumina
Umay1	SRR12990819	PRJNA674756	*Ustilago maydis*	Fungi	Illumina
Umay2	SRR12990820	PRJNA674756	*Ustilago maydis*	Fungi	Illumina
Umay3	SRR12990821	PRJNA674756	*Ustilago maydis*	Fungi	Illumina
Athal1	SRR21871460	PRJNA887677	*Arabidopsis thaliana*	Plantae	Illumina

Athal2	SRR21871466	PRJNA887677	*Arabidopsis thaliana*	Plantae	Illumina
Athal3	SRR21871467	PRJNA887677	*Arabidopsis thaliana*	Plantae	Illumina
Csativus1	SRR25709139	PRJNA1007410	*Cucumis sativus*	Plantae	Illumina
Csativus2	SRR25709141	PRJNA1007410	*Cucumis sativus*	Plantae	Illumina
Csativus3	SRR25709138	PRJNA1007410	*Cucumis sativus*	Plantae	Illumina
Otauri1	ERR5417781	PRJEB43294	*Ostreococcus tauri*	Protista	Illumina
Otauri2	ERR5417780	PRJEB43294	*Ostreococcus tauri*	Protista	Illumina
Otauri3	ERR5417779	PRJEB43294	*Ostreococcus tauri*	Protista	Illumina
Plas1	SRR32867107	PRJNA1242290	*Plasmodium vivax*	Protista	Illumina
Plas2	SRR32867100	PRJNA1242290	*Plasmodium vivax*	Protista	Illumina
Plas3	SRR32867098	PRJNA1242290	*Plasmodium vivax*	Protista	Illumina
Pcap1	SRR13176614	PRJNA481983	*Phytophthora capsici*	Stramenopila	Illumina
Pcap2	SRR8259121	PRJNA505815	*Phytophthora capsici*	Stramenopila	Illumina
Pcap3	SRR5944957	PRJNA386483	*Phytophthora capsici*	Stramenopila	Illumina
Pram1	SRR16939097	PRJNA780196	*Phytophthora ramorum*	Stramenopila	Illumina
Pram2	SRR16939100	PRJNA780196	*Phytophthora ramorum*	Stramenopila	Illumina
Pram3	SRR16939107	PRJNA780196	*Phytophthora ramorum*	Stramenopila	Illumina