Build a Bioinformatic Analysis Platform and Apply it to Routine Analysis of Microbial Genomics and Comparative Genomics

Hualin Liu
Huazhong Agricultural University https://orcid.org/0000-0002-3630-5522
Bingyue Xin
Huazhong Agricultural University
Jinshui Zheng (jszheng@mail.hzau.edu.cn)
Huazhong Agricultural University https://orcid.org/0000-0002-0147-5900
Hao Zhong
Huazhong Agricultural University
Yun Yu
Huazhong Agricultural University
Donghai Peng
Huazhong Agricultural University
Ming Sun (m98sun@mail.hzau.edu.cn)
Huazhong Agricultural University https://orcid.org/0000-0001-5465-5983

Method Article

Keywords: comparative genomics, COG annotation, phylogenetic orthology, phylogenetic analysis, variants calling, pan-genome, genome assembly, gene prediction, genome annotation, genome distance, Average Nucleotide Identity, PGCGAP

Posted Date: April 25th, 2022

DOI: https://doi.org/10.21203/rs.2.21224/v6

Abstract

More and more frequently, genomics and comparative genomics have been used as routine methods for general microbiological research. However, using several tools or even writing some scripts are required for completing a simple analysis, which is complicated for most biological researchers. To simplify the operation process, particularly for the convenience of microbiologists, here we have developed PGCGAP, a comprehensive, malleable, and easily installed prokaryotic genomic and comparative genomic analysis pipeline. PGCGAP implements genome assembly, gene prediction and annotation, genome and metagenome distance estimation, phylogenetic analysis, COG annotation, pan-genome analysis, inference of orthologous gene groups, variant calling and annotation, and screening for antimicrobial and virulence genes. Although we have tried our best to simplify the installation and usage of PGCGAP, it may be difficult for non-bioinformaticians to master it. Therefore, a protocol was created to help microbiologists without any experience in bioinformatics to establish their bioinformatics platform and perform routine analyses. This protocol shows how to choose the equipment to install a Linux subsystem on a laptop with a Windows 10 system, to install the PGCGAP and perform all analyses with an example dataset (**This protocol applies to PGCGAP version 1.0.34 and later**). The protocol requires a basic understanding of Linux, so an additional web page was written to help uninitiated users learn Linux and whole-genome sequencing (https://github.com/liaochenlanruo/pgcgap/wiki/Learning-bioinformatics or http://bcam.hzau.edu.cn/linuxwgs.php).

Introduction

Genome sequencing has become a routine method for common microbiological studies owing to the steady decrease in the cost of genome sequencing. Various tools have been developed for genome analysis. However, for general users, it takes time to install and learn to use various programs and prepare the related input files. Even for some simple objectives, users need to spend much effort to integrate several tools or even write scripts. For example, when we need a core-genome-SNP-based phylogenetic analysis for isolates from the same species, we should successively use the following tools, Bowtie2[1] or BWA[2] for read mapping, Samtools[3] or GATK[4] for SNP calling, and FastTree[5] or RAxML[6] for phylogenetic tree construction. Therefore, a comprehensive, flexible, and efficient pipeline for general analysis is urgently needed. We developed a prokaryotic genomic and comparative genomic analysis pipeline named PGCGAP to coordinate several genomic analysis software packages and in-house scripts to meet the various needs of microbiologists.

Development of the protocol

PGCGAP was developed to facilitate the work of genomic and comparative genomic analyses of microbes. Considering the important role of basic bioinformatics in microbial research and that most microbiologists lack bioinformatic analysis skills, this protocol describes in detail the installation of Linux systems and demonstrates software installation methods. Finally, we demonstrated systematically all the applications of PGCGAP using the example datasets.

Applications of the protocol

PGCGAP can be used for (i) reads preprocess and genome assembly, (ii) gene prediction and annotation, (iii) genome and metagenome distance estimation, (iv) phylogenetic analysis, (v) COG annotation, (vi) pan-genome analysis, (vii) inference of orthologous gene groups, (viii) variant calling and annotation, and (ix) screening for antimicrobial and virulence genes. Noteworthy, although the entire pipeline was developed for prokaryotes, some modules, such as “Assemble”, “MASH”, “OrthoF”, “CoreTree”, “AntiRes”, and “STREE”, can also be used for the analysis of eukaryotic genomes. Moreover, the “VAR” module can be applied for the analysis of any haploid genome.

Advantages and limitations of this pipeline

PGCGAP is versatile, feature-rich, easy to install and use, and friendly to microbiologists and bioinformatics beginners. New features will continue to be added.

Expertise required to implement the protocol

Users need to be skilled in using computers, and it will be easier to master this protocol if they have some Linux skills. A webpage introducing the basics of Linux, usage of common commands, software installation, and whole-genome sequencing technology was developed to help users get started with bioinformatics. Please visit https://github.com/liaochenlanruo/pgcgap/wiki/Learning-bioinformatics or http://bcam.hzau.edu.cn/linuxwgs.php for more information.

Overview of the procedure

Twelve frequently used prokaryotic genomic and comparative genomic analysis processes were integrated into PGCGAP as different modules. Modules can be used separately or in different combinations for various purposes (Fig. 1). (i) “Assemble” performs genome assembly of Illumina reads, third-generation reads, and hybrid reads using ABySS[7], SPAdes[8], Canu[9], and Unicycler[10]. The paired-end reads will be preprocessed with Fastp[11] to remove adapters, polyG tail, and low-quality reads before genome assembly. (ii) “Annotate” performs gene prediction and genome annotation by Prokka[12]. (iii) “ANI” computes Average Nucleotide Identity (ANI) between each genome pair by fastANI[13]. Three scripts “triangle2list.pl”, “get_ANImatrix.pl”, and “Plot_ANIheatmap.R” have been developed here to generate the ANI matrix and plot the correlation matrix heat map (Supplementary Figure S1). (iv) “MASH” estimates genome and metagenome distance and similarity using MinHash[14], and a heat map of genome similarity will be generated by two scripts “get_Mash_Matrix.pl” and “Plot_MashHeatmap.R”. (v) “Pan” calls Roary[15] to calculate the pan-genome. Two scripts “fmplot.py” and “plot_3Dpie.R” were developed for result visualization (Supplementary Figure S2). A phylogenetic tree based on single-copy core proteins called by Roary[15] will be constructed (Supplementary Figure S3). (vi) COG (Clusters of Orthologous Group) annotation can be conducted using the module “pCOG”. Amino acid sequences of each genome are aligned against the COG database by DIAMONG[16], and then all of the best hits will be mapped to the COG functional category by in-house scripts. The R script “Plot_COG.R” was written for result visualization

(Supplementary Figure S4). Comparison and visualization of COG functional categories among different genomes can be done by Perl script “get_flag_relative_abundances_table.pl” and R script “Plot_COG_Abundance.R” (Supplementary Figure S5). (vii) “OrthoF” uses OrthoFinder[17] for phylogenetic orthology inference. Gene duplication events are also predicted (Supplementary Figure S6). A phylogenetic tree based on single-copy orthologue protein sequences will be constructed. (viii) “CoreTree” was developed for genome-wide phylogenetic analysis based on the protein sequences or SNPs of single-copy core genes. First, CD-HIT[18] is used to rapidly generate protein clusters, and then the protein sequences of single-copy core genes are extracted by Perl scripts and aligned using MAFFT[19]. Second, on the one hand, alignments of protein sequences are concatenated, and the phylogenetic tree with the best model is constructed by IQ-TREE[20] (Supplementary Figure S7). Contrarily, the protein sequence alignments are converted into corresponding codon alignments using PAL2NAL v14[21]. Then, the codon alignments are concatenated, and SNP-sites[22] are called to find the SNP sites. Finally, IQ-TREE[20] is used to construct the SNP phylogenetic tree (Supplementary Figure S8). (ix) “AntiRes” calls abricate[23] to screen for antimicrobial and virulence genes from contigs. (x) “VAR” performs genome-wide variants calling by mapping methods. First, single-end or paired-end reads are mapped to a reference genome by BWA[2] after filtering by Sickle[24]. Second, variant calling and annotation are performed by Freebayes[25] and snpEff[26], respectively. Then, the whole-genome SNP alignment and core SNP alignment are obtained using the snippy-core[27] software and SNP-sites, respectively. Then the phylogenetic tree of core SNP alignments will be constructed by IQ-TREE. (xi) “STREE” constructs a phylogenetic tree based on multiple FASTA sequences in one file. First, the sequences are aligned by MUSCLE[28], and then trimAL[29] is used to obtain the conserved blocks of aligned sequences. Finally, IQ-TREE is used for phylogenomic inference. (xii) “ACC” integrates other useful gadgets and now includes the function “Assess” for filtering short sequences in the genome and assessing the status of the genome only.

Experimental design

Selection of reference genome format for variants calling

The reference genome can be files in FASTA and GenBank formats. If a GenBank file rather than a FASTA file was supplied as the reference, annotation information of the variants was generated to show the user whose features were affected by the variants.

How to balance speed and assembly quality when assembling Illumina reads

From our experience, ABySS[7] can complete genome assembly faster and with fewer computer resources. The assembly qualities of Unicycler[10] and SPAdes[8] are better than ABySS[7], occupy more computer resources and run very slowly. Therefore, we strongly recommend that users choose the auto mode for Illumina data assembly. PGCGAP first calls ABySS[7] for Illumina read assembly. When the N50 of the assembled genome is less than 50,000, it automatically calls Unicycler[10] and SPAdes[8] to try multiple parameters for another assembly.

Choice of a module to calculate pairwise genome distance

Both “ANI” and “MASH” can calculate the pairwise genome distance. “MASH” is more suitable for dealing with thousands of genomes as it runs faster. In addition to nucleotide sequences and assembled genomes, “MASH” can also take amino acid sequences and raw sequencing reads as inputs, and can be used to calculate distances between metagenomic samples. It is worth noting that no ANI output will be reported if the ANI value is below 78 %, and in this case, MASH can be used instead.

Reagents

No reagents are required for this protocol. But example datasets were needed to practice the application of PGCGAP. The example datasets used in this protocol can be downloaded at http://bcam.hzau.edu.cn/PGCGAP/PGCGAP_Examples.tar.gz.

Equipment

A laptop, desktop PC, or server can be used to build a bioinformatic analysis platform, and the suggested hardware requirements are listed in Table 1. Slightly lower features are also allowed (CPU must have four logical processors, memory must be greater than 8 G), but the computing speed may decrease, and the capacity of the hard disk can be adjusted according to actual requirements.

Procedure

Building a bioinformatic analysis platform on Windows 10

The Windows Subsystem for Linux (WSL) allows users to install Linux subsystems directly on a Windows 10 system. It can easily run Linux commands and install Linux software to avoid the installation of third-party virtual machine software. The advantage of WSL is that it makes better use of computer memory and does not require copying files between the host and the virtual machine.

Configuration of WSL

Timing ~1 min

System requirements: Windows 10 Version 1709, Build 16299, or above 64-bit systems.

1. Enable WSL: Open “Settings”, click “Apps”, then find and click “Programs and Features”, click “Turn Windows features on or off”, find “Windows Subsystem for Linux” and check the box, click “OK”, and restart the computer (Supplementary Video 1).

Install Linux

Timing ~59 min

2. Open the Microsoft Store, search Ubuntu, and choose to install Ubuntu 18.04 LTS. Follow the prompts to set up your username and password. Here, we create an account with the username “bio” (Supplementary Video 2). When the installation is finished, we need to do some configuration on the system (Supplementary Video 3).

3. Enter the following command in the terminal to update the source:

$sudo apt-get update

4. Set the password for root.

$sudo passwd root

5. Enable the CUDA-aware MPI.

For Linux 64, the CUDA awareness support may be disabled by default. Users should enable the support by setting the environment variable to use OpenMPI. Check whether CUDA awareness support is enabled in the environment variable configuration file (~/.bashrc). If it is not enabled, enter the following commands in the terminal.

$echo OMPI_MCA_opal_cuda_support=true >> ~/.bashrc

$source ~/.bashrc

6. Installation of Miniconda

(A) Installation of Miniconda on Linux

(i) Here, Miniconda will be installed; go to the official website, and select the installation file suitable for your system and Python version (Supplementary Video 4).

$wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

(ii) Start installation

$bash Miniconda3-latest-Linux-x86_64.sh

Keep pressing “Enter” key when prompted to visualize the license agreement, enter “yes” and press “Enter” to continue. Press “Enter” to confirm the default installation location. Miniconda is installed in the miniconda3 directory, under the user’s home directory. Type “yes” and press “Enter” to initialize miniconda3. Finally, type the command “source ~/.bashrc” in the terminal.

$source ~/.bashrc

(iii) Set up the Bioconda channel. Add the channels by entering the following three commands in the terminal.

$conda config --add channels defaults

$conda config --add channels bioconda

$conda config --add channels conda-forge

(B) Install Miniconda on MacOS

(i) Installation of Miniconda3

$wget https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh

$sh Miniconda3-latest-MacOSX-x86_64.sh

$source ~/.bash_profile

(ii) Add channels of Bioconda

$conda config --add channels defaults

$conda config --add channels bioconda

$conda config --add channels conda-forge

Installation of PGCGAP (Supplementary Video 5).

Timing ~34 min

7. Create a pgcgap environment and install PGCGAP (Usually specify the latest version of PGCGAP).

$conda install -y mamba

$mamba create -n pgcgap pgcgap=1.0.34 java-jdk=8.0.112

8. Activate the pgcgap environment.

$conda activate pgcgap

9. Set up the COG database.

$pgcgap --setup-COGdb

10. Exit the pgcgap environment.

$conda deactivate

Step by Step examples

Timing ~20 h

The usage and parameters of PGCGAP can be viewed by typing “pgcgap -h” in the terminal. Next, we show how to run all the modules of the PGCGAP through a dataset.

11. Download and decompress the example dataset.

$wget http://bcam.hzau.edu.cn/PGCGAP/PGCGAP_Examples.tar.gz

$tar -zxvf PGCGAP_Examples.tar.gz

In this example, the working directory is located at the H drive. All hard disks in Windows were mounted in the “/mnt” directory of Ubuntu Linux. The “PGCGAP_Examples/Reads/Illumia” directory contains six Illumina Hiseq paired-end reads of *Escherichia coli*; the “PGCGAP_Examples/Reads/Oxford” directory contains the Oxford Nanopore reads of *Escherichia coli* K12; and the “PGCGAP_Examples/Reads/PacBio” directory contains the Pacific Biosciences released P6-C4 chemistry reads of *Escherichia coli* K12. “PGCGAP_Examples/Reads/MG1655.gbff” is the GenBank format file of *E. coli* K-12 *substr*. MG1655, and will be used as the reference genome. The “PGCGAP_Examples/Reads/Hybrid” directory contains two short reads files and one long reads file of the same strain. “PGCGAP_Examples/Other_inputs/ proteins.fas” contains 18 protein sequences of MFS transporter from several bacterial species.

12. Activate the pgcgap environment.

$conda activate pgcgap

13. Example 1: Genome assembly with Illumina reads.

Paired-end reads of six strains in the directory “Reads/Illumina/” are used as inputs. In the dataset, the naming format of the genome is “strain_1.fastq.gz, ” and “strain_2.fastq.gz”. The string after the strain name is “_1.fastq.gz”, and its length is 11, so “--suffix_len” was set to 11. Users can choose “abyss”, “spades”, and “auto” for genome assembly. The assembly speed with “abyss” is faster, and the assembly quality with “spades” is better. Taking into account the speed and quality of assembly, we suggest using the “auto” mode for assembly. “--filter_length” is set here to remove sequences shorter than 200 bp from the assembled genomes.

$pgcgap --Assemble --platform illumina --assembler abyss --filter_length 200 --ReadsPath Reads/Illumina --reads1 _1.fastq.gz --reads2 _2.fastq.gz --kmmer 81 --threads 4 --suffix_len 11

$pgcgap --Assemble --platform illumina --assembler spades --filter_length 200 --ReadsPath Reads/Illumina --reads1 _1.fastq.gz --reads2 _2.fastq.gz --threads 4 --suffix_len 11

$pgcgap --Assemble --platform illumina --assembler auto --filter_length 200 --ReadsPath Reads/Illumina --reads1 _1.fastq.gz --reads2 _2.fastq.gz --kmmer 81 --threads 4 --suffix_len 11

New directories and documents are generated after the program is completed. The assembly results for each genome are in the “Results/Assembles/Illumina” directory, while all scaffolds of the strains are stored in “Results/Assembles/Scaf/Illumina”. “*.filtered.fas” is the genome with short sequences removed. “*.prefilter.stats” describes the status of the genome before filtering, and “*.filtered.stats” describes the status of the genome after short sequence filtering. While “abyss” was chosen as the assembler, users are advised to check the assembly stats file (such as Results/Assembles/Illumina/SRR9620252_assembly/SRR9620252-stats.tab) of each genome to ensure that the value of N50 is greater than 50,000 bp. The file “scaf.list” under the working directory contains the absolute path of all genomes, which will be used as the input file of module ANI.

14. Example 2: ONT reads assembly and polishing.

The Oxford nanopore produces only one read file (“Reads/Oxford/oxford.fasta”), so only the parameter of “–reads1” needs to be set. Here, the value “.fasta”. “–genomeSize” is the estimated genome size, and users can check the genome size of similar strains in the NCBI database for reference. The parameter is set to “4.8m”. The suffix of the reads file here is “.fasta” and its length is 6, so “–suffix_len” is set to 6.

$pgcgap –Assemble –platform oxford –ReadsPath Reads/Oxford –reads1 .fasta –genomeSize 4.8m –threads 4 –suffix_len 6 –filter_length 200

The results are stored in the “Results/Assembles/Oxford” and “Results/Assembles/Scaf/Oxford” directories. The former contains all intermediate files and genome files, while the latter contains only the assembled genome.

15. Example 3: PacBio reads assembly and polishing.

PacBio also produces only one read file (“Reads/PacBio/pacbio.fastq”); the parameter settings are similar to those of Oxford. The strain name is “pacbio” with the suffix “.fastq” and the suffix length is 6, so “–suffix_len” was set to 6.

$pgcgap –Assemble –platform pacbio –ReadsPath Reads/PacBio –reads1 .fastq –genomeSize 4.8m –threads 4 –suffix_len 6 –filter_length 200

The results are stored in the “Results/Assembles/PacBio” and “Results/Assembles/Scaf/PacBio” directories. The former contains all intermediate files and genome files, while the latter contains only the assembled genome.

16. Example 4: hybrid assembly of short reads and long reads and polishing.

Paired-end short reads and long reads in the directory “Reads/Hybrid/” are used as inputs. Illumina reads and long reads had been obtained from the same isolates.

$pgcgap –Assemble –platform hybrid –ReadsPath Reads/Hybrid –short1 short_reads_1.fastq.gz –short2 short_reads_2.fastq.gz –long long_reads_high_depth.fastq.gz –threads 4

The results are stored in the “Results/Assembles/Hybrid” directory, and the final assembly is named “assembly.fasta”.

17. Example 5: Gene prediction and annotation.

Here, the assembly results of Illumina reads are taken as inputs (“Results/Assembles/Scaf/Illumina/*.filtered.fas”). The suffix of the genome is “.filtered.fas”. When running the program, the value of the “–Scaf_suffix” parameter cannot be quoted. Here, .filtered.fas should not be quoted.

$pgcgap –Annotate –scafPath Results/Assembles/Scaf/Illumina –Scaf_suffix .filtered.fas –genus Escherichia –species coli –codon 11 –threads 4

The generated files are stored in the “Results/Annotations” directory, and files in the directories “Results/Annotations/AAs”, “Results/Annotations/CDs” and “Results/Annotations/GFF” will be used for subsequent analysis.

18. Example 6: Constructing the single-copy core protein tree and core SNP tree.

The phylogenetic trees of single-copy core proteins and single-copy core gene SNPs will be constructed using the six *E. coli* genomes sequenced by Illumina as datasets. The input files are the amino acid sequence files (“Results/Annotations/AAs/*.faa”) and the nucleotide sequence files (“Results/Annotations/CDs/*.ffn”) obtained by genome annotation. Amino acid files and nucleotide files must be suffixed with “.faa” and “.ffn”, respectively. The “.faa” and “.ffn” files of the same strain should have the same prefix name. The name of protein IDs and gene IDs in the amino acid file and nucleotide file should start with the strain name. The Prokka[12] software was suggested to generate the input files. “strain_num” is a very important parameter to specify the number of genomes for analysis. Three parameters including –fasttree, –bsnum (default), and –fastboot are provided to choose the method for phylogenetic tree constructing.

$pgcgap –CoreTree –CDsPath Results/Annotations/CDs –AAsPath Results/Annotations/AAs –codon 11 –strain_num 6 –threads 4

The result files are stored in the directory “Results/CoreTrees”. “ALL.core.protein.fasta.gb.treefile” and “ALL.core.snp.fasta.gb.treefile” are the phylogenetic tree files constructed based on the aligned sequences of the single-copy core proteins and the core SNPs with the best-fit model of evolution, respectively. The best-fit model of evolution for the protein alignments and SNPs alignments can be found in the file “ALL.core.protein.fasta.gb.iqtree” and “ALL.core.snp.fasta.gb.iqtree”, respectively. While –fasttree was sprcified, “ALL.core.protein.nwk” will be outputted as the phylogenetic tree of single-copy core proteins. Users can import tree files into MEGA[30] or iTOL[31] to view the topology.

19. Example 7: Constructing the single-copy core protein tree only.

If the “--CDsPath” was set to “NO”, the nucleotide files will not be needed, and the phylogenetic tree of core SNPs will not be constructed.

$pgcgap --CoreTree --CDsPath NO --AAsPath Results/Annotations/AAs --codon 11 --strain_num 6 --threads 4

20. Example 8: pan-genome analysis and phylogenetic tree construction.

(A) GFF3, also known as generic feature format version 3, is a tab-delimited, plain text file used to describe features of DNA, RNA, and protein sequences. For Pan-genome analysis, GFF3 files (With “.gff” as the suffix) of each strain are placed into a directory (“Results/Annotations/GFF/*.gff”). We strongly recommend using Prokka[12] to generate the aforementioned files. If the Annotate module runs first, the files are automatically generated. Users can set the minimum percentage identity for blastp by the parameter “identi”, and only an integer number is allowed. Users can choose any of the three parameters including --fasttree, --bsnum (default), and --fastboot to construct the phylogenetic tree.

$pgcgap --Pan --codon 11 --strain_num 6 --threads 4 --GffPath Results/Annotations/GFF --PanTree --AAsPath Results/Annotations/AAs

The results are stored in the “Results/PanGenome” directory. A spreadsheet named “gene_presence_absence.csv” lists each gene and which samples contained it. Users can take the gene_presence_absence.csv file and a trait file to conduct pan-genome wide association studies with the Scoary[34] software. At the same time, some visual results (“*.pdf”) are also outputted. A phylogenetic tree based on the aligned sequences of single-copy core proteins called by Roary will be constructed automatically if the parameter “PanTree” was provided. The tree file can be found in the directory “Results/PanGenome/Core/” with a name of “Roary.core.protein.fasta.gb.treefile” or “Roary.core.protein.fasta.gb.nwk”. The best-fit model of evolution for the protein alignments can be found in the file “Roary.core.protein.fasta.gb.iqtree”.

21. Example 9: Inference of orthologous gene groups.

The input files are also the amino acid sequence files suffixed with “.faa” (“Results/Annotations/AAs/*.faa”). Users can choose any of the three parameters including --fasttree, --bsnum (default), and --fastboot to construct the phylogenetic tree.

$pgcgap --OrthoF --threads 4 --AAsPath Results/Annotations/AAs

The resulting files are placed in the “Results/OrthoFinder/Results_orthoF” directory. The file “Single.Copy.Orthologue.fasta.gb.treefile” in directory “Single_Copy_Orthologue_Tree” is the tree file of single-copy orthologues, and the best fit module can be found in the file “Single.Copy.Orthologue.fasta.gb.iqtree” under the same directory.

22. Example 10: Compute whole-genome Average Nucleotide Identity.

The input file named “scaf.list” contains the absolute path of each genome, one per line. If the “--Assemble” function is run first, the list file is generated automatically. The value of the parameter “--Scaf_suffix” depends on the actual situation, here is “.filtered.fas”.

$pgcgap --ANI --threads 4 --queryL scaf.list --refL scaf.list --ANIO Results/ANI/ANIs --Scaf_suffix .filtered.fas

The results are stored in the “Results/ANI” directory. The file “ANI” contains comparison information of genome pairs. The document is composed of five columns, each of which represents the query genome, reference genome, ANI value, count of bidirectional fragment mappings, and total query fragments. A heat map file “ANI_matrix.pdf” is generated.

23. Example 11: Genome and metagenome similarity estimation using MinHash

This requires genome files (complete or draft) in a directory as inputs (Default: Results/Assembles/Scaf/Illumina).

$pgcgap --MASH --scafPath Results/Assembles/Scaf/Illumina --Scaf_suffix .filtered.fas

The results are stored in the “Results/MASH” directory. The file “MASH” shows the pairwise distance between pair genomes, and each column represents Reference-ID, Query-ID, Mash-distance, P-value, and Matching-hashes. A heat map file named “MASH_matrix.pdf” is generated to describe the similarity of each genome pair.

24. Example 12: COG annotation.

The input files are also the amino acid sequence files suffixed with “.faa” (“Results/Annotations/AAs/*.faa”).

$pgcgap --pCOG --threads 4 --strain_num 6 --AAsPath Results/Annotations/AAs

The results are stored in the “Results/COG” directory. The super COG table of each strain (“*.2Scog.table”) and its plot (“*.2Scog.table.pdf”) will be generated. “All_flags_relative_abundances.table” is a table containing the relative abundance of each flag for all strains, while “All_flags_relative_abundances.pdf” is the corresponding visualization result.

25. Example 13: Variants calling and phylogenetic tree construction based on a reference genome.

The six genomes sequenced by Illumina were chosen as datasets (“Reads/Illumina/*.gz”). *Escherichia coli* K-12 *substr*. MG1655 was selected as the reference genome and the reference file “MG1655.gbff” in the GenBank format is stored in the “Reads” directory. The absolute path of the reference genome (here is “/mnt/h/PGCGAP_Examples/Reads/MG1655.gbff”) is required to run the program.

$pgcgap --VAR --threads 4 --refgbk /mnt/h/PGCGAP_Examples/Reads/MG1655.gbff --ReadsPath Reads/Illumina --reads1 _1.fastq.gz --reads2 _2.fastq.gz --suffix_len 11 --strain_num 6 --qualtype sanger

The resulting files are stored in the “Results/Variants” directory, where the “Core” directory contains the core SNPs of all strains and their phylogenetic tree.

26. Example 14: Screening of contigs for antimicrobial and virulence genes

This requires genome files (complete or draft) in a directory as inputs (Default: Results/Assembles/Scaf/Illumina). Users can choose one of the following databases for analysis: argannot[32], card[33], ecoh[34], ecoli_vf (https://github.com/phac-nml/ecoli_vf), ncbi[35], plasmidfinder[36], resfinder[37], MEGARes[38], and vfdb[39]. Users can set the value of parameter “db” to “all” to search against to all of the databases.

$pgcgap –AntiRes –scafPath Results/Assembles/Scaf/Illumina –Scaf_suffix .filtered.fas –threads 4 –db all –identity 75 –coverage 50

The resulting files are stored in the “Results/AntiRes” directory. “*.tab” files are screening results of each strain, and the “summary.txt” file contains a matrix of gene presence/absence for all strains.

27. Example 15: Perform all analyses for paired-end reads.

In fact, we provide many one-line commands to complete analyses of each module with original reads, which have been demonstrated in the examples of module Annotate, pCOG, CoreTree, Pan, OrthoF, ANI, MASH, and AntiRes. In addition, users can complete almost all analyses with original reads by just one line of commands through the module All. In short, the module All will call modules Assemble, Annotate, pCOG, CoreTree, Pan, OrthoF, ANI, MASH, and AntiRes one by one to perform the analyses. For the sake of flexibility, the module VAR was not integrated into module All and needed to be added when needed. Only the reads and reference files should be provided, and important parameters should also be provided.

$pgcgap –All –platform illumina –filter_length 200 –ReadsPath Reads/Illumina –reads1 _1.fastq.gz –reads2 _2.fastq.gz –suffix_len 11 –kmmer 81 –genus Escherichia –species coli –codon 11 –strain_num 6 –threads 4 –VAR –refgbk /mnt/h/PGCGAP_Examples/Reads/MG1655.gbff –qualtype sanger –PanTree

28. Example 16: Filter short sequences in the genome and assess the status of the genome.

“Assess” takes the assembled genomes as inputs. First, it assesses the stats of the genome; second, the sequences shorter than “–filter_length” are deleted from the genome. Finally, the stats of the filtered genome are assessed. The results files are stored in the same directory as the inputs.

$pgcgap –ACC –Assess –scafPath Results/Assembles/Scaf/Illumina –Scaf_suffix -8.fa –filter_length 200

29. Example 17: Construct a phylogenetic tree based on multiple FASTA sequences in one file.

“STREE” takes the file containing multiple-FASTA sequences as input. The parameter “–bsnum” represents the number of bootstraps.

$pgcgap –STREE –seqfile Other_inputs/proteins.fas –seqtype p –bsnum 500 –threads 4

The result files will be stored in the directory “Results/STREE”. The file “proteins.fas.aln.gb.treefile” is the final phylogenetic tree file. The best-fit model of evolution for the protein alignments can be found in the file “proteins.fas.aln.gb.iqtree”.

Troubleshooting

Troubleshooting advice can be found in Table 2.

Time Taken

The following marked time was tested in the WSL on the laptop. The features of the laptop were as follows: i7-4710MQ CPU (with 4 cores and 8 logical processors), 16 GB DDR3L RAM, 240 G SSD, and 1 T HDD. All commands are called 4 threads.

Step 1: configuration of WSL, 1 min.

Step 2: installation of Linux, 43 min.

Step 3-5: configuration of Linux, 10 min.

Step 6: installation of Miniconda, 6 min.

Step 7-10: installation of PGCGAP, 34 min.

Step 11: download and decompress example datasets, 11 min.

Step 12: activate the pgcgap environment, 8 s.

Step 13: Illumina reads assembly by abyss, spades, and auto, 40 min, 2 h, and 1 h, respectively.

Step 14: ONT reads assembly, 1 h.

Step 15: PacBio reads assembly, 44 min.

Step 16: hybrid assembly of short reads and long reads, 18 min

Step 17: gene prediction and annotation, 26 min.

Step 18: constructing the single-copy core protein tree and core SNP tree, 1.5 h.

Step 19: constructing the single-copy core protein tree only, 1.2 h.

Step 20: pan-genome analysis and phylogenetic tree constructing, 1 h.

Step 21: inference of orthologous gene groups, 2 h.

Step 22: compute whole-genome Average Nucleotide Identity, 19 s.

Step 23: genome similarity estimation using MinHash, 7 s.

Step 24: COG annotation, 13 min.

Step 25: variant calling, and phylogenetic tree construction based on the reference genome, 6.7 h.

Step 26: Screening of contigs for antimicrobial and virulence genes, 4 min.

Step 27: Perform all functions for paired-end reads, 13 h.

Step 28: Filter short sequences in the genome and assess the status of the genome, 15 s.

Step 29: Construct a phylogenetic tree based on multiple FASTA sequences in one file, 1 h.

Anticipated Results

The output files of example datasets by PGCGAP can be downloaded at http://bcam.hzau.edu.cn/PGCGAP/PGCGAP_Results.tar.gz.

References

1 Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**, 357 (2012).

2 Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv:1303.3997v2 [q-bio.GN]* (2013).

3 Heng Li *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)* **25**, 2078-2079 (2009).

4 McKenna, A. *et al.* The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.* **20**, 1297-1303 (2010).

5 Price, M. N., Dehal, P. S. & Arkin, A. P. FastTree 2–approximately maximum-likelihood trees for large alignments. *PLoS One* **5**, e9490 (2010).

6 Stamatakis, A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* **30**, 1312-1313 (2014).

7 Jackman, S. D. *et al.* ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter. *Genome Res.* **27**, 768-777 (2017).

8 Nurk, S. *et al.* Assembling Genomes and Mini-metagenomes from Highly Chimeric Reads. 158-170 (Springer Berlin Heidelberg).

9 Koren, S. *et al.* Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res.* **27**, 722-736 (2017).

10 Wick, R. R., Judd, L. M., Gorrie, C. L. & Holt, K. E. Unicycler: Resolving bacterial genome assemblies from short and long sequencing reads. *PLoS Comp. Biol.* **13**, e1005595 (2017).

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

12 Seemann, T. Prokka: rapid prokaryotic genome annotation. *Bioinformatics* **30**, 2068-2069 (2014).

13 Jain, C., Rodriguez-R, L. M., Phillippy, A. M., Konstantinidis, K. T. & Aluru, S. High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nat Commun* **9**, 5114-5114 (2018).

14 Ondov, B. D. *et al.* Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology* **17**, 132 (2016).

15 Page, A. J. *et al.* Roary: rapid large-scale prokaryote pan genome analysis. *Bioinformatics* **31**, 3691-3693 (2015).

16 Buchfink, B., Reuter, K. & Drost, H.-G. Sensitive protein alignments at tree-of-life scale using DIAMOND. *Nat Methods*. **18**, 366-368 (2021).

17 Emms, D. M. & Kelly, S. OrthoFinder: phylogenetic orthology inference for comparative genomics. *Genome Biology* **20**, 238 (2019).

18 Li, W. & Godzik, A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* **22**, 1658-1659 (2006).

19 Katoh, K., Misawa, K., Kuma, K. & Miyata, T. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.* **30**, 3059-3066 (2002).

20 Minh, B. Q. *et al.* IQ-TREE 2: New Models and Efficient Methods for Phylogenetic Inference in the Genomic Era. *Mol Biol Evol*. **37**, 1530-1534 (2020).

21 Suyama, M., Torrents, D. & Bork, P. PAL2NAL: robust conversion of protein sequence alignments into the corresponding codon alignments. *Nucleic Acids Res.* **34**, W609-612 (2006).

22 Page, A. J. *et al.* SNP-sites: rapid efficient extraction of SNPs from multi-FASTA alignments. *Microb Genom* **2**, e000056 (2016).

23 Seemann, T. Abricate, Github https://github.com/tseemann/abricate.

24 Joshi NA & JN, F. Sickle: A sliding-window, adaptive, quality-based trimming tool for FastQ files (Version 1.33) [Software]. Available at https://github.com/najoshi/sickle. (2011).

25 Garrison E & Marth G. Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907 [q-bio.GN]* (2012).

26 Cingolani, P. *et al.* A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3. *Fly (Austin)* **6**, 80-92 (2012).

27 Seemann, T. Snippy: Rapid haploid variant calling and core genome alignment. Available at https://github.com/tseemann/snippy. (2014).

28 Edgar, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32**, 1792-1797 (2004).

29 Capella-Gutiérrez, S., Silla-Martínez, J. M. & Gabaldón, T. trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses. *Bioinformatics*. **25**, 1972-1973 (2009).

30 Kumar, S., Stecher, G., Li, M., Knyaz, C. & Tamura, K. MEGA X: Molecular Evolutionary Genetics Analysis across Computing Platforms. *Mol. Biol. Evol.* **35**, 1547-1549 (2018).

31 Letunic, I. & Bork, P. Interactive tree of life (iTOL) v3: an online tool for the display and annotation of phylogenetic and other trees. *Nucleic Acids Res.* **44**, W242-W245 (2016).

32 Gupta, S. K. *et al.* ARG-ANNOT, a new bioinformatic tool to discover antibiotic resistance genes in bacterial genomes. *Antimicrob. Agents Chemother.* **58**, 212-220 (2014).

33 Jia, B. *et al.* CARD 2017: expansion and model-centric curation of the comprehensive antibiotic resistance database. *Nucleic Acids Res.* **45**, D566-d573 (2017).

34 Ingle, D. J. *et al.* In silico serotyping of E. coli from short read data identifies limited novel O-loci but extensive diversity of O:H serotype combinations within and between pathogenic lineages. *Microbial genomics* **2**, e000064-e000064 (2016).

35 Feldgarden, M. *et al.* Validating the AMRFinder Tool and Resistance Gene Database by Using Antimicrobial Resistance Genotype-Phenotype Correlations in a Collection of Isolates. *Antimicrobial agents and chemotherapy* **63**, e00483-00419 (2019).

36 Carattoli, A. *et al.* In silico detection and typing of plasmids using PlasmidFinder and plasmid multilocus sequence typing. *Antimicrob. Agents Chemother.* **58**, 3895-3903 (2014).

37 Zankari, E. *et al.* Identification of acquired antimicrobial resistance genes. *J. Antimicrob. Chemother.* **67**, 2640-2644 (2012).

38 Doster, E. *et al.* MEGARes 2.0: a database for classification of antimicrobial drug, biocide and metal resistance determinants in metagenomic sequence data. *Nucleic Acids Res*. **48**, D561-D569 (2020).

39 Chen, L., Zheng, D., Liu, B., Yang, J. & Jin, Q. VFDB 2016: hierarchical and refined dataset for big data analysis–10 years on. *Nucleic Acids Res.* **44**, D694-697 (2016).

Acknowledgements

This work was made possible through funding from the National Key R&D Program of China (2017YFD0201201), National Natural Science Foundation of China (31670085, 31970003, 31770003 and 31700002), and China 948 Program of Ministry of Agriculture (2016-X21). H.L. and B.X. contributed equally to this work. Author order was determined by their seniority and initial contribution.

Figures

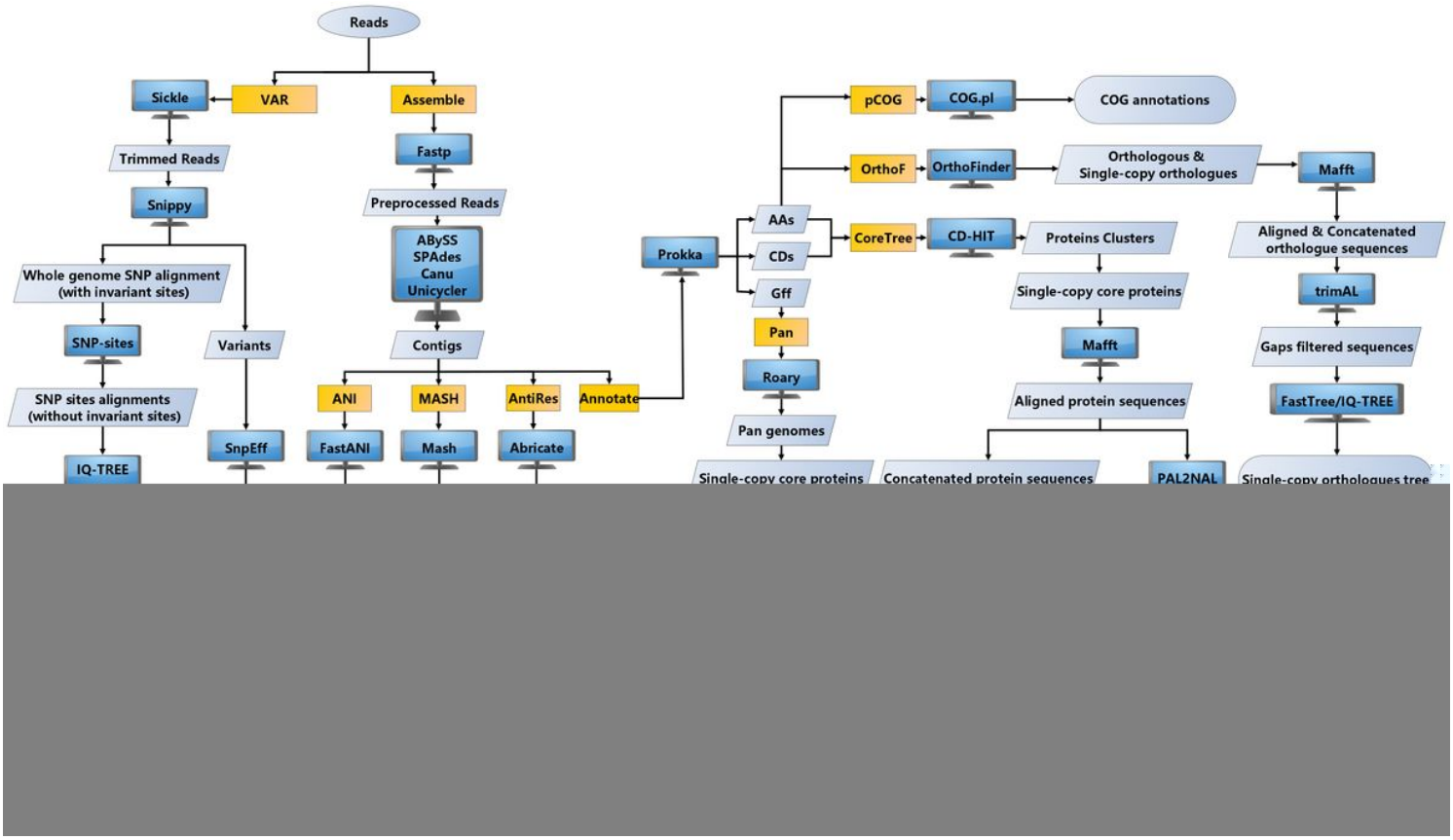

Figure 1

Ten frequently used prokaryotic genomics and comparative genomics analysis processes were integrated into PGCGAP as different modules. Modules can be used separately or in combinations for various purposes.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplementaryFigureS3.Aphylogenetictreeofsinglecopycoreproteinscalledby Pan.jpg
- SupplementaryFigureS2.Plotsofscriptsfmplot.pyandplot3Dpie.R.jpg
- SupplementaryFigureS1.Thecorrelationmatrixheatmapdrawnbyfunction ANIofPGCGAP.jpg
- SupplementaryFigureS5.Aheatmapdepictstherelativeabundanceofeachflagforallstrains.jpg
- SupplementaryFigureS6.Arootedspeciestreeforthespeciesbeinganalyzedinferredby OrthoF.jpg
- SupplementaryFigureS7.PhylogenetictreeofsinglecopycoreproteinsgeneratedbymoduleCoreTree.jpg
- SupplementaryFigureS8.SNPsphylogenetictreeofSinglecopycoregenesgeneratedbymoduleCoreTree.jpg
- SupplementaryVideo1EnableWSL.mp4
- SupplementaryVideo2InstallUbuntu.mp4
- SupplementaryVideo4InstallBioconda.mp4
- SupplementaryVideo5InstallPGCGAP.mp4
- Table1.Suggestedhardwarerequirementsforabioinformaticsanalysisplatform.docx
- SupplementaryVideo3ConfigureUbuntu.mp4
- SupplementaryFigureS4.Apicturedescribesthefrequencyofeachflagforthestrain SRR9620252.jpg
- Table2.Troubleshooting.docx