Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Build a Bioinformatics 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, antimicrobial and virulence genes mining, PGCGAP

DOI: https://doi.org/10.21203/rs.2.21224/v3

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Abstract

Genomics and comparative genomics have been increasingly used as routine methods for general microbiological researches. However, it is usually necessary to call several tools or even write some scripts to complete some simple analysis, which is complicated for most biological researchers. To simplify the operation process, especially for the convenience of microbiologists in the analysis, here we have developed PGCGAP, a comprehensive, malleable and easily-installed prokaryotic genomics and comparative genomics analysis pipeline, which implements genome assembly, gene prediction and annotation, average nucleotide identity (ANI) calculation, phylogenetic analysis, COG annotation, pan-genome analysis, inference of orthologous gene groups, variants 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-bioinformatician users to master it. So, a protocol was created to help microbiologists without any experience in bioinformatics to establish their own bioinformatics platform and perform routine analysis. This protocol shows how to choose equipment, to install a Linux subsystem on a laptop with windows 10 system, to install PGCGAP and perform all analysis with an example dataset. 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).

Introduction

Genome sequencing has become a routine method for common microbiological studies as continuously decrease in the cost of genome sequencing. Various tools have been developed for genome analysis. But for general users, it takes time to install and learn to use various programs and prepare the related input files. Even for some simple purposes, users need to spend much effort to integrate several tools or even write some scripts. For example, when we need a core-genome-SNP based phylogenetic analysis for the isolates from same species, we should successively use the following tools, Bowtie2[1] or BWA[2] for reads 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 genomics and comparative genomics 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 genomics and comparative genomics analysis of microbes. Considering the important role of basic bioinformatics in microbial research and most microbiologists lacking analysis skills, this protocol describes in detail the installation of Linux systems and demonstrates the software installation methods. Finally, we demonstrated all the usages of PGCGAP step by step through the example datasets.

Applications of the protocol

PGCGAP can be used for (i) genome assembly, (ii) gene prediction and annotation, (iii) genome distance estimation, (iv) phylogenetic analysis, (v) COG annotation, (vi) pan-genome analysis, (vii) inference of

orthologous gene groups, (viii) variants calling and annotation and (ix) screening for antimicrobial and virulence genes. It is worth noting that although the entire pipeline was developed for prokaryotes, some of the modules such as “Assemble”, “MASH”, “OrthoF”, “CoreTree”, “AntiRes” and “STREE” can also be used for the analysis of eukaryotic genomes. Besides, “VAR” applies to 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. But a Graphical User Interface (GUI) has not been developed.

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 the whole-genome sequencing technology was developed to help users get started with bioinformatics. Please visit https://github.com/liaochenlanruo/pgcgap/wiki/Learning-bioinformatics for more information.

Overview of the procedure

Twelve frequently used prokaryotic genomics and comparative genomics 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, the third generation reads, and the hybrid reads using ABySS[7], SPAdes[8], Canu[9], and Unicycler[10], respectively. 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), respectively. (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” (Supplementary Figure S2). (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 S3). A phylogenetic tree based on single-copy core proteins called by Roary[17] will be constructed (Supplementary Figure S4). (vi) COG (Clusters of Orthologous Group) annotation was conducted by module “pCOG”. Amino acid sequences of each genome were blasted against the COG database, and then all hits were mapped to the COG functional category by in-house scripts. R script “Plot_COG.R” was written for result visualization (Supplementary Figure S5). 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 S6). (vii) “OrthoF” uses OrthoFinder[16] for phylogenetic orthology inference. Gene duplication events will be also predicted (Supplementary Figure S7). (viii) “CoreTree” was developed for genome-wide phylogenetic analysis based on the protein sequences or SNPs of single-copy core genes. Firstly, CD-HIT[17] was used to rapidly generate protein clusters, and then the protein sequences of single-copy core genes were extracted by Perl

scripts and aligned by MAFFT[18]. Secondly, on one hand, alignments of protein sequences were concatenated, and the phylogenetic tree was constructed by ModelTest-NG[19] and RAxML-NG[20] (Supplementary Figure S8). On the other hand, the protein sequence alignments were converted into corresponding codon alignments by PAL2NAL v14[21]. Then, the codon alignments were concatenated, and SNP-sites[22] was called to find SNP sites. Finally, ModelTest-NG[19] and RAxML-NG[20] were used to construct the SNPs phylogenetic tree (Supplementary Figure S9). (ix) “AntiRes” calls abricate[23] to screen for antimicrobial and virulence genes from contigs. (x) “VAR” performs genome-wide variants calling by mapping methods. Firstly, single-end or paired-end reads were mapped to a reference genome by BWA[2] after filtered by Sickle[24]. Secondly, variants calling and annotation were performed by Freebayes[25] and snpEff[26], respectively. Then, the whole genome SNP alignment and core SNP alignment were obtained by snippy-core[27]. Finally, Gubbins[28] was used to remove SNPs influenced by recombination events of the whole genome SNP alignment and construct a phylogenetic tree (Supplementary Figure S10). Furthermore, the phylogenetic tree of core SNP alignment was constructed by ModelTest-NG[19] and RAxML-NG[20] (Supplementary Figure S11). (xi) “STREE” Construct a phylogenetic tree based on multiple-FASTA sequences in one file. Firstly, the sequences were aligned by MUSCLE[29], and then Gblocks[30] was used to get the conserved blocks of the aligned sequences. Finally, IQ-TREE[31] was used for phylogenomic inference (Supplementary Figure S12). (xii) “ACC” integrated other useful gadgets, and it now includes 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 format and GenBank format. If a GenBank file rather than a FASTA file was supplied as the reference, annotation information of the variants will be generated to show the user which feature was affected by the variants.

How to balance speed and assembly quality when assembling Illumina reads

According to our experience, ABySS[7] can complete genome assembly faster with fewer computer resources. While the assembly quality of Unicycler[10] and SPAdes[8] is better than ABySS[7], but it occupies more computer resources and runs very slow. Therefore, we strongly recommend that users choose the auto mode for Illumina data assembly. PGCGAP calls ABySS[7] for Illumina reads assembly firstly. When 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 construct the phylogenetic tree of single-copy core proteins

Both “CoreTree” and “Pan” can be used to construct a phylogenetic tree of single-copy core proteins. That which module should be used depends on which type of input file the user has. “CoreTree” takes only the amino acid sequence files as inputs, while “Pan” needs both amino acid sequence files and Gff3 files. Besides, according to the default threshold, the number of single-copy core proteins obtained by “CoreTree” and “Pan”

may be different. Users can choose the module which generates more single-copy core proteins to build the phylogenetic tree.

Choice of a module to calculate pairwise genome distance

Both “ANI” and “MASH” can calculate pairwise genome distance. “MASH” is more suitable for dealing with thousands of genomes because of its faster running speed. 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.

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://122.205.95.26/PGCGAP/PGCGAP_Examples.tar.gz.

Equipment

A laptop, desktop PC or server can be used to build a bioinformatics 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 bioinformatics analysis platform on Windows 10

Windows Subsystem for Linux (WSL) allows users to install Linux subsystems directly on Windows 10 system. It can easily run Linux commands and install Linux software, avoiding 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**). After 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. If it is not enabled, enter the following command 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. Here, Miniconda 3 will be installed (**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

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

$source ~/.bashrc

(iii) Set up 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 for the installation of PGCGAP.

$conda create -n pgcgap python=3.6

8. Activate the pgcgap environment.

$conda activate pgcgap

9. Installation of PGCGAP.

$conda install pgcgap

10. Check if the dependent software packages were installed.

$pgcgap --check-external-programs

11. Set up the COG database.

$pgcgap --setup-COGdb

12. Exit the pgcgap environment.

$conda deactivate

Step by Step examples

Timing ~2.3 d

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 PGCGAP through a dataset.

13. 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 locates 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” is the GenBank format file of *E.coli* K-12 *substr.* MG1655, 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 bacteria species.

14. Activate the pgcgap environment.

$conda activate pgcgap

15. Example 1: Genome assembly with Illumina reads.

Paired-end reads of six strains in the directory “Reads/Illumina/” were 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 the genome assembly. The assembly speed with “abyss” is faster, and the assembly quality with “spades” is better. To take into account the speed and quality of assembly, we suggest using “auto” mode for assembly. “–filter_length” was 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 were generated after the program is finished. The assembly results for each genome are in the “Results/Assembles/Illumina” directory, and all scaffolds of the strains were 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 sequences 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.

16. Example 2: Oxford reads assembly.

Oxford nanopore only produces one reads file (“Reads/Oxford/oxford.fasta”), so only the parameter of “--reads1” needs to be set, here the value is “.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 was set to “4.8m” here. The suffix of the reads file here is “.fasta” and its length is 6, so “--suffix_len” was 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” directory and the “Results/Assembles/Scaf/Oxford” directory. The former contains all intermediate files and genome files, the latter contains only the assembled genome.

17. Example 3: PacBio reads assembly.

PacBio also produces only one reads file too (“Reads/PacBio/pacbio.fastq”), the parameter settings are similar to 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” directory and in the “Results/Assembles/Scaf/PacBio” directory. The former contains all intermediate files and genome files, the latter containing only the assembled genome.

18. Example 4: hybrid assembly of short reads and long reads.

Paired-end short reads and long reads in the directory “Reads/Hybrid/” were used as inputs. Illumina reads and long reads must be 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 was named as “assembly.fasta”.

19. Example 5: Gene prediction and annotation.

Here, the assembly results of Illumina reads were taken as inputs (“Results/Assembles/Scaf/Illumina/*.fa”). The suffix of the genome is “-8.fa”. When running the program, the value of the “–Scaf_suffix” parameter cannot be quoted. Here, -8.fa should not be quoted.

$pgcgap –Annotate –scafPath Results/Assembles/Scaf/Illumina –Scaf_suffix -8.fa –genus Escherichia –species “Escherichia 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.

20. Example 6: Constructing the single-copy core proteins tree and core SNPs tree.

The phylogenetic trees of single-copy core proteins and single-copy core genes 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 the 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 acids file and nucleotide file should be started with the strain name. The Prokka[12] software was suggested to generate the input files.

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

The result files are stored in the “Results/CoreTrees” directory. “ALL.core.protein.*.support” and “ALL.core.snp.*.support” are the phylogenetic tree files of the single-copy core proteins and the core SNPs constructed with the best-fit model of evolution, respectively. Users can import these two files into MEGA[32] or iTOL[33] to view the topology.

21. 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 too.

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

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

GFF3 files (With “.gff” as the suffix) of each strain placed into a directory (“Results/Annotations/GFF/*.gff”). They must contain the nucleotide sequence at the end of the file. Protein sequence files (one per species) in FASTA format under another directory were also needed (“Results/Annotations/AAs/*.faa”) if the parameter “–PanTree” was provided for constructing a phylogenetic tree. It should be noted that the “*.gff” file and the

“*.faa” file must correspond. We strongly recommend using Prokka[12] to generate the files. If the “–Annotate” function was run first, the files will be generated automatically.

$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 sample is presented in. Users can take the gene_presence_absence.csv file and a traits file to conduct pan-genome wide association studies with Scoary[34] software. At the same time, some visual results (“*.pdf”) are also outputted. “Results/PanGenome/Core/Roary.core.protein.BIC.AIC.AICc.HIVW+I+G4+F.raxml.support” is the phylogenetic tree constructed based on the single-copy core proteins called by Roary[15] software. “HIVW+I+G4+F” represent the best-fit model of evolution for the protein alignments according to AIC[35], AICc, and BIC[36] statistical criteria. If the parameters “–PanTree” and “–AAsPath” were not provided, the phylogenetic tree will not be constructed.

23. Example 9: Inference of orthologous gene groups.

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

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

The resulting files are placed in the “Results/OrthoFinder/Results_orthoF” directory.

24. 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 was run first, the list file will be generated automatically. The value of the parameter “–Scaf_suffix” depends on the actual situation, here is “-8.fa”.

$pgcgap –ANI –threads 4 –queryL scaf.list –refL scaf.list –ANIO Results/ANI/ANIs –Scaf_suffix -8.fa

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 ANI value, count of bidirectional fragment mappings and total query fragments, respectively. A heat map file “ANI_matrix.pdf” was generated.

25. Example 11: Genome similarity estimation using MinHash

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

$pgcgap –MASH –scafPath Results/Assembles/Scaf/Illumina –Scaf_suffix -8.fa

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

26. 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, and "All_flags_relative_abundances.pdf" is the corresponding visualization result.

27. 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"). Taking *Escherichia coli* K-12 *substr*. MG1655 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.

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

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

$pgcgap --AntiRes --scafPath Results/Assembles/Scaf/Illumina --Scaf_suffix -8.fa --threads 4 --db ncbi --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.

29. Example 15: Perform all functions for paired-end reads.

Only the reads file and reference file should be provided. For the sake of flexibility, the "VAR" function needs to be added extra.

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

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

"Assess" takes the assembled genomes as inputs. Firstly, it assesses the stats of the genome; Secondly, the sequences shorter than "--filter_length" were deleted from the genome; Finally, the stats of the filtered genome

was assessed. The results files will be stored in the same directory as the inputs.

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

31. 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 results files will be stored in “Results/STREE”. The file “proteins.fas.aln.gb.treefile” contains the final phylogenetic tree.

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

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 are as follows: i7-4710MQ CPU (with 4 cores and 8 logical processors), 16 GB DDR3L RAM, 240 G SSD, 1 T HDD. All commands called 4 threads.

Step 1, configuration of WSL, 1 min.

Step 2, installation of Linux, 43 min.

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

Step 6, installation of Miniconda, 6 min.

Step 7-12, installation of PGCGAP, 34 min.

Step 13, download and decompress the example dataset, 11 min.

Step 14, activate the pgcgap environment, 8 s.

Step 15, Illumina reads assembly by abyss, spades and auto, 43 min, 12.8 h, and 5.7 h, respectively.

Step 16, Oxford reads assembly, 1.6 h.

Step 17, PacBio reads assembly, 54 min.

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

Step 19, gene prediction and annotation, 1 h.

Step 20, constructing the single-copy core protein tree and core SNPs tree, 2.6 h.

Step 21, constructing the single-copy core protein tree only, 2.4 h.

Step 22, pan-genome analysis and phylogenetic tree constructing, 3 h.

Step 23, inference of orthologous gene groups, 51 min.

Step 24, compute whole-genome Average Nucleotide Identity, 17 s.

Step 25, genome similarity estimation using MinHash, 1 min.

Step 26, COG annotation, 20.3 h.

Step 27, variant calling, and phylogenetic tree construction based on the reference genome, 11.8 h.

Step 28, Screening of contigs for antimicrobial and virulence genes, 30 s.

Step 29, Perform all functions for paired-end reads, 1.9 d.

Step 30, Filter short sequences in the genome and assess the status of the genome, 15 s.

Step 31, Construct a phylogenetic tree based on multiple-FASTA sequences in one file, 3 h.

Anticipated Results

The output files of example datasets by PGCGAP can be downloaded at http://122.205.95.26/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 Emms, D. M. & Kelly, S. OrthoFinder: phylogenetic orthology inference for comparative genomics. *Genome Biology* **20**, 238 (2019).

17 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).

18 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).

19 Darriba, D. *et al.* ModelTest-NG: A New and Scalable Tool for the Selection of DNA and Protein Evolutionary Models. *Mol. Biol. Evol.* **37**, 291-294 (2019).

20 Kozlov, A. M., Darriba, D., Flouri, T., Morel, B. & Stamatakis, A. RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. *Bioinformatics* **35**, 4453-4455 (2019).

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 Croucher, N. J. *et al.* Rapid phylogenetic analysis of large samples of recombinant bacterial whole genome sequences using Gubbins. *Nucleic Acids Res.* **43**, e15 (2015).

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

30 Castresana, J. Selection of Conserved Blocks from Multiple Alignments for Their Use in Phylogenetic Analysis. *Mol. Biol. Evol.* **17**, 540-552 (2000).

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

32 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).

33 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).

34 Brynildsrud, O., Bohlin, J., Scheffer, L. & Eldholm, V. Rapid scoring of genes in microbial pan-genome-wide association studies with Scoary. *Genome Biology* **17**, 238 (2016).

35 Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* **19**, 716-723 (1974).

36 Schwarz, G. Estimating the Dimension of a Model. *Ann. Statist.* **6**, 461-464 (1978).

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 and 31770003), and China 948 Program of Ministry of Agriculture (2016-X21).

Figures

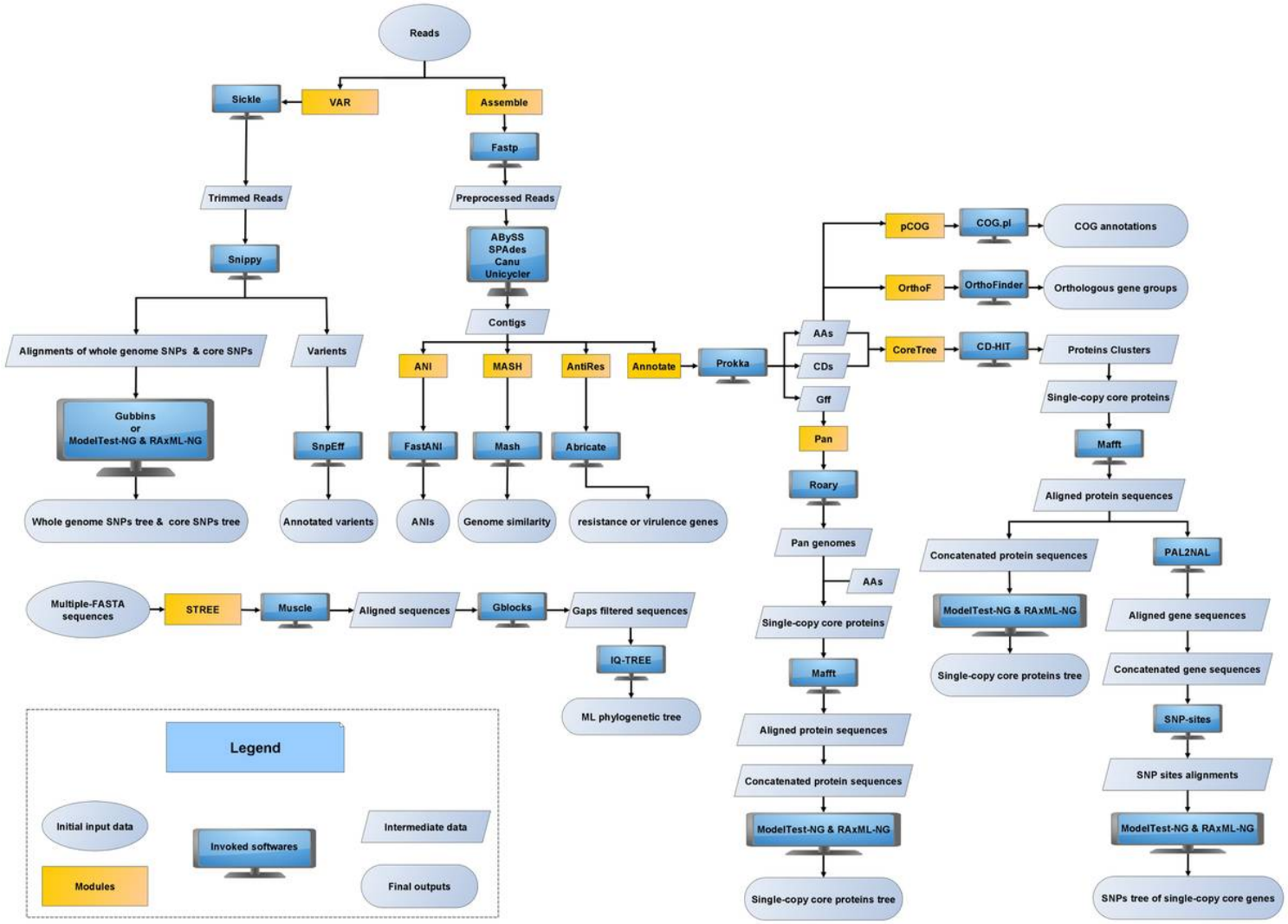

Figure 1

The Module “ACC” was not described in the diagram.

Supplementary Files

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

- SupplementaryFigureS1.ThecorrelationmatrixheatmapdrawnbymoduleANIofPGCGAP.jpg
- SupplementaryFigureS2.ThecorrelationmatrixheatmapdrawnbythemoduleMASHofPGCGAP.jpg
- SupplementaryVideo1EnableWSL.mp4
- SupplementaryFigureS3.Plotsofscriptsfmplot.pyandplot3Dpie.R.jpg
- SupplementaryVideo2InstallUbuntu.mp4
- SupplementaryFigureS4.Aphylogenetictreeofsinglecopycoreproteinscalledbymodule Pan.jpg
- SupplementaryFigureS5.Apicturedescribesthefrequencyofeachflagforthestrain SRR9620252.jpg
- SupplementaryVideo4InstallBioconda.mp4
- SupplementaryVideo5InstallPGCGAP.mp4

- SupplementaryFigureS8.PhylogenetictreeofsinglecopycoreproteinsgeneratedbymoduleCoreTree.jpg
- SupplementaryFigureS7.ArootedspeciestreeforthespeciesbeinganalyzedinferredbymoduleOrthoF.jpg
- Table1.Suggestedhardwarerequirementsforabioinformaticsanalysisplatform.docx
- SupplementaryFigureS6.AheatmapdepictstherelativeabundanceofeachflagforallstrainsS.jpg
- SupplementaryFigureS10.AphylogenetictreeofthewholegenomeSNPalignmentgeneratedbymoduleVAR.jpg
- SupplementaryFigureS11.AphylogenetictreeofthecoreSNPalignmentgeneratedbymoduleVAR.jpg
- SupplementaryFigureS9.SNPsphylogenetictreeofSinglecopycoregenesgeneratedbymoduleCoreTre.jpg
- SupplementaryVideo3ConfigureUbuntu.mp4
- Table2.Troubleshooting.docx
- SupplementaryFigureS12.AphylogenetictreeoftheMFStransportergeneratedbymoduleSTREE.jpg