Oct 21, 2024

Decoding Prolonged COVID-19: From Cluster Analysis to Pathway Mapping

DOI
dx.doi.org/10.17504/protocols.io.81wgbzm51gpk/v1

Spoorthi R Kulkarni[1], Lavanya C[1], Vidya Niranjan[1]

[1]R V College of Engineering

Centre of Excellence in ...

Vidya Niranjan
R V College of Engineering

DOI: https://dx.doi.org/10.17504/protocols.io.81wgbzm51gpk/v1

Protocol Citation: Spoorthi R Kulkarni, Lavanya C, Vidya Niranjan 2024. Decoding Prolonged COVID-19: From Cluster Analysis to Pathway Mapping. **protocols.io https://dx.doi.org/10.17504/protocols.io.81wgbzm51gpk/v1**

Protocol status: Working
We use this protocol and it's working

Created: October 18, 2024

Last Modified: October 21, 2024

Protocol Integer ID: 110281

Keywords: Prolong covid, SNP, Consensus, Pathway, Clustering , clusters against the reference genome, clc genomics workbench, reference genome, prolonged covid, using clc genomics workbench, cluster analysis to pathway mapping, subsequent blast analysis, identifying key gene, multiple sequence alignment, pathway analysis, key gene, multiple sequence alignment with mafft, pathway mapping, integrating population genetics, specific genetic variation, uncovering pathway, population genetics

Abstract

This protocol outlines a comprehensive approach to decoding prolonged COVID by integrating population genetics and pathway analysis, providing a framework for predicting geographic regions, identifying key genes, and uncovering pathways that can inform targeted treatments. Initially, sequences obtained from NCBI are subjected to phylogenetic and clustering analysis using CLC Genomics Workbench, followed by multiple sequence alignment with MAFFT. The reference genome is indexed, and sequences are aligned to produce SAM and BAM files, leading to variant calling through Freebayes. Consensus sequences are generated using tools like bcftools, and subsequent BLAST analysis is performed to compare clusters against the reference genome. Python scripts further analyze SNP variations, generating Excel reports that highlight significant differences. The workflow offers insights into geographic-specific genetic variations, helping to reveal pathways involved in prolonged COVID, which could ultimately aid in the development of more effective, regionally-tailored treatments.

Guidelines

Ensure all required software tools (e.g., CLC Genomics Workbench, MAFFT, BWA, Samtools, Freebayes, BCFTools, BLAST, Python) are installed and functional.
Install necessary Python libraries if any scripts are included, and verify script compatibility with your system.

Troubleshooting

Before start

- Ensure that all required tools and databases are properly downloaded, installed, and configured before starting the analysis.
- Verify that the reference genome file for the selected organism is correctly indexed and stored in the designated reference folder for easy access during the analysis.
- Confirm that the input sequence files are organized and named appropriately to avoid confusion during processing.
- Check the system environment for compatibility with the software tools being used, ensuring that dependencies are installed.
- Backup important files and outputs regularly to prevent data loss during the workflow.
- Review the parameters for each tool to ensure they align with the objectives of the study, making any necessary adjustments to optimize performance.

To retrieve SarsCov2 virus genome sequences reported in India from the NCBI Virus database

1 Access to NCBI Virus Database (**https://www.ncbi.nlm.nih.gov/labs/virus/**)

- Once on the results page, locate the **"Filters"** section on the left-hand side.

Dataset

Sample and reference genome retrieved from NCBI virus

https://www.ncbi.nlm.nih.gov/labs/virus/vssi/#/virus?SeqType_s=Nucleotide&VirusLineage_ss=Severe%20acute%20respiratory%20syndror

- Under the "Geographic region" filter, select **"India"** as the geographic location.

1.1 Download the data, use the "**Download**" button on the top right corner of the page and 4,827 genome sequences were retrieved.

Multiple Sequence Alignment

2 The multiple sequence alignment is performed using MAFFT version 7 tool. The MAFFT version 7 tool efficiently aligned the 4,827 sequences using a fast and scalable algorithm, producing a high-quality multiple sequence alignment. Based on this alignment, a phylogenetic tree was constructed using the maximum likelihood method, revealing evolutionary relationships between the sequences.

Software

MAFFT NAME

Cluster analysis

3 Using **CLC Genomics Workbench**, a phylogenetic tree was generated from 4,827 aligned sequences utilizing the tool's built-in maximum likelihood algorithm. Based on

the tree and a similarity matrix, genomic clusters were formed by grouping sequences with high similarity. A total of 50 distinct clusters were extracted, each representing a set of closely related sequences. These clusters were identified directly from the tree, enabling efficient classification and comparative analysis of the genomic data.

Software

CLC Genomics Workbench	NAME
windows 11	OS
Qiagen	DEVELOPER

Alignment and VCF Generation

4 Indexing a reference genome is a critical preparatory step in genomic analysis, particularly for tasks such as read alignment and variant calling. The process involves creating an indexed version of the reference genome, which allows for faster and more efficient search and retrieval operations during sequence alignment. The reference genome, typically in FASTA format, is processed by a genome analysis tool such as BWA to build indexed files.

Command

Indexing the reference genome (ubuntu 20.04)

Indexing the reference genome

```
bwa index Reference_genome.fasta
```

5 The alignment of sequencing reads to the reference genome was performed using the **BWA-MEM** algorithm. In this step, the indexed reference genome (Reference_genome.fasta) served as the target for alignment, while the input sample file (sample_ID.fasta) contained the reads to be aligned. The BWA MEM algorithm, optimized for long reads, efficiently mapped the reads to the reference genome. The alignment results were saved in SAM format (sample_ID.sam), which provides detailed information on the mapping of each read, including its position and quality scores. This alignment serves as the foundation for subsequent analyses, such as variant detection and genome annotation.

Command

Alignment (ubuntu 20.04)

Alignment

```
bwa mem Reference_genome.fasta sample_ID.fasta > sample_ID.sam
```

6 The SAM file generated from the alignment was converted to the more efficient BAM format using **Samtools.** This step compressed and converted the human-readable **SAM** file into a binary **BAM** file, which is much smaller in size and faster to process. The -@ 4 option was used to enable multi-threading with 4 CPU cores, speeding up the conversion. The resulting sample_ID.bam file retains all the alignment information and is now ready for further analysis, such as sorting, indexing, and variant calling.

Command

Sam to Bam conversion (ubuntu 20.04)

Sam to Bam conversion

```
samtools view -@ 4 -Sb -o sample_ID.bam sample_ID.sam
```

7 To identify genetic variants from the aligned sequencing data, the **FreeBayes** variant caller was employed. In this process, the reference genome was specified with the -f option, indicating the FASTA file (reference.fasta) that serves as the baseline for variant detection. The input file, input.bam, contained the aligned sequencing reads in BAM format. FreeBayes then scanned these reads against the reference genome to detect variants, including single nucleotide polymorphisms (SNPs) and small insertions or deletions (indels). The identified variants were outputted in a **VCF** (Variant Call Format) file named variants.vcf, which includes essential details such as genomic positions, reference and alternate alleles, and quality metrics. This VCF file provides a structured representation of the genetic variation in the analyzed sample.

Software

freebayes	NAME
Garrison and Marth	DEVELOPER
https://github.com/freebayes/freebayes	SOURCE LINK

Command

Generation of VCF using Freebayes

Generation of VCF using Freebayes

```
freebayes -f reference.fasta input.bam > variants.vcf
```

Consensus Sequence Generation

8 The process of consensus sequence generation involves using information from Variant Call Format (VCF) files to create a sequence that reflects the specific genetic variants of a sample compared to a reference genome. The VCF file is initially compressed and indexed for efficient access, while the reference genome is similarly indexed to enable rapid retrieval of specific regions. Once prepared, the variants from the VCF are applied to the reference genome, resulting in a consensus sequence that represents the genetic profile of the sample. This process not only allows for the reconstruction of the sample's genomic sequence but also helps in identifying and reducing the dataset to the most unique Single Nucleotide Polymorphisms (SNPs). By focusing on these consensus SNPs.

Command

Zipping the VCF files

Zipping the VCF files

```
bgzip filename.vcf
```

Command

new command name

```
tabix filename.vcf
```

The command samtools faidx reference_seq.fasta MN908947.3 | bcftools consensus C1.vcf.gz > C1.fa generates a consensus sequence for a specific cluster, referred to as C1 in this case. **samtools faidx reference_seq.fasta, MN908947.3:** This part of the command uses samtools to index and retrieve the reference sequence (reference_seq.fasta), specifically the region corresponding to MN908947.3 (the SARS-CoV-2 reference genome). The tool allows efficient access to any part of the reference genome, which is essential for generating consensus sequences based on specific variants. **bcftools consensus C1.vcf.gz:** The bcftools consensus command takes the compressed and indexed VCF file (C1.vcf.gz), which contains the identified variants for cluster C1. This VCF represents the genetic differences between the samples in cluster C1 and the reference genome. **> C1.fa:** The output from the bcftools consensus process is directed into a file named C1.fa, which represents the consensus sequence. This file contains the reference genome sequence modified with the specific variants from the VCF file for cluster C1.

Command

Consensus SNP generation (ubuntu 20.04)

Consensus SNP generation

```
samtools faidx reference_seq.fasta MN908947.3 | bcftools consensus
C1.vcf.gz > C1.fa
```

Conducting a BLAST analysis using the consensus SNPs and the reference genome

9 A BLAST search was performed to compare consensus SNP sequences against the reference genome The **query file** (C1.fa) contained the consensus SNPs, while the **subject file** (reference_seq.fasta) represented the reference genome. The **blastn** algorithm aligned the SNP sequences to the reference, generating detailed output including query and subject sequence IDs, percentage identity, alignment length, mismatches, gap openings, and start-end positions in both the query and reference sequences. Additionally, E-values, bit scores, and the aligned sequences were included in the output. The results were saved in C1_vs_refseq.out.

Command

Blastn

Blastn

```
blastn -query C1.fa -subject reference_seq.fasta -outfmt "6 qseqid
sseqid pident length mismatch gapopen qstart qend sstart send evalue
bitscore qseq sseq" -out C1_vs_refseq.out
```

Execute Python code to generate C_vs_ref_SNPs.xlsx

10 This Python code parses BLAST output files and identifies single nucleotide polymorphisms (SNPs) by comparing query sequences with a reference genome. The script reads a BLAST output file (in tab-delimited format), extracts relevant columns (such as query and subject sequences, alignment positions, and sequence mismatches), and compares the sequences to detect SNPs. For each SNP, it records details like the position, reference base, and variant base, which are stored in a list. The collected SNP data is then converted into a **pandas** DataFrame, and the results are saved as an Excel file (_SNPs.xlsx). The script processes multiple files (e.g., C1_vs_ref.out to C56_vs_ref.out) in sequence, generating an Excel file for each.

Command

Python code to convert C1_vs_Refseq.out to an Excel file named C1_vs_ref_SNPs.xlsx

Python code to convert C1_vs_Refseq.out to an Excel file named C1_vs_ref_SNPs.xlsx

```
import pandas as pd

def parse_blast_output(blast_output_file):
    # List to store SNP information
    snp_data = []

    with open(blast_output_file) as f:
        for line in f:
            # Skip header lines or empty lines
            if line.startswith("#") or not line.strip():
                continue

            cols = line.strip().split("\t")
            print(f"Number of columns: {len(cols)}")
            print(f"Columns: {cols}")

            # Check if the line has at least 13 columns (including
sequences)
            if len(cols) < 13:
                print(f"Skipping incomplete line: {line}")
                continue

            try:
                query_id = cols[0]
                subject_id = cols[1]
                percent_identity = float(cols[2])
                alignment_length = int(cols[3])
                mismatches = int(cols[4])
                gap_opens = int(cols[5])
                query_start = int(cols[6])
                query_end = int(cols[7])
                subject_start = int(cols[8])
                subject_end = int(cols[9])
                e_value = float(cols[10])
                bit_score = float(cols[11])
```

```
                query_seq = cols[12]
                subject_seq = cols[13]

                # Compare sequences and identify SNPs
                for i in range(min(len(query_seq), len(subject_seq))):
                    if query_seq[i] != subject_seq[i]:
                        query_pos = query_start + i
                        subject_pos = subject_start + i
                        snp = {
                            'Query ID': query_id,
                            'Subject ID': subject_id,
                            'Position': subject_pos,
                            'Reference Base': subject_seq[i],
                            'Variant Base': query_seq[i]
                        }
                        snp_data.append(snp)
                        print(f"SNP found at position {subject_pos}:
{subject_seq[i]} -> {query_seq[i]}")
            except IndexError as e:
                print(f"Error processing line: {line}")
                print(f"Exception: {e}")
            except ValueError as e:
                print(f"Error converting to integer: {line}")
                print(f"Exception: {e}")

    # Convert the SNP data to a DataFrame
    df = pd.DataFrame(snp_data)
    # Create an output filename based on the input filename
    output_filename = blast_output_file.replace(".out", "_SNPs.xlsx")
    # Save the DataFrame to an Excel file
    df.to_excel(output_filename, index=False)

# Example usage
for i in range(1,57):
    parse_blast_output(f"C{i}_vs_ref.out")
    i += 1
```

Gene Identification for Detected SNPs

11 The **Ensembl COVID-19 Tool** was utilized to analyze the genes associated with the consensus SNPs produced from the BLAST alignment. This tool maps the SNP positions to known genomic regions, allowing the identification of genes or regulatory

elements affected by the variants. By inputting the consensus SNPs, the tool links each variant to its corresponding gene, providing insights into potential functional impacts. This gene annotation step is critical for understanding how the detected SNPs might influence biological processes or disease mechanisms, particularly in the context of COVID-19 research.

Software

ensembl-vep NAME

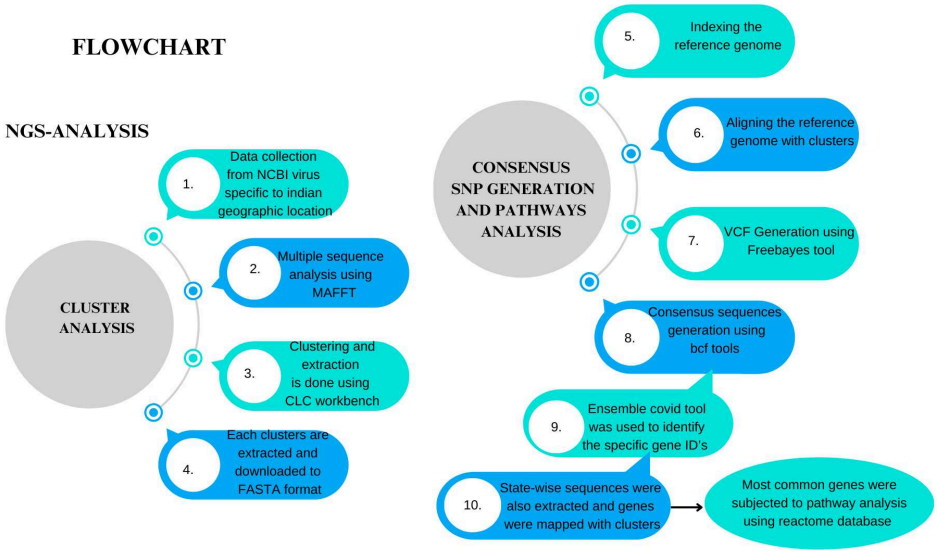
Identification of pathways related to the corresponding genes

12 The identified genes were analyzed using the **Reactome database**, which facilitated the mapping of these genes to specific biological pathways. By inputting the gene data into Reactome, pathways related to the genes were identified, particularly those associated with prolonged COVID-19 symptoms. This analysis helped to highlight key molecular processes and pathways that may be impacted by the detected variants, offering insights into the mechanisms underlying long-term effects of COVID-19 and potential therapeutic targets.

https://reactome.org/PathwayBrowser/

Pathway name	Curated found	Curated Total	Interactor found	Interactor Total	Entities found	Entities Total	Entities ratio	Entities pValue	Entities FDR	Reactions found	Reac tot
Gene and protein expression by JAK-STAT signaling after Interleukin-12 stimulation	0	73	1	688	1	752	0.031	6.17E-2	6.17E-2	1	
G alpha (q) signalling events	0	285	1	484	1	753	0.031	6.17E-2	6.17E-2	1	
Biological oxidations	0	552	1	231	1	773	0.032	6.33E-2	6.33E-2	5	
Metabolism of vitamins and cofactors	0	395	1	412	1	786	0.033	6.44E-2	6.44E-2	1	
Translation of Structural Proteins	1	81	1	757	1	812	0.034	6.65E-2	6.65E-2	12	
VEGFA-VEGFR2 Pathway	0	129	1	767	1	843	0.035	6.9E-2	6.9E-2	1	
Fatty acid metabolism	0	428	1	438	1	856	0.036	7E-2	7E-2	1	
SARS-CoV-2-host interactions	1	248	0	690	1	867	0.036	7.09E-2	7.09E-2	2	
Apoptotic cleavage of cellular proteins	0	38	1	851	1	868	0.036	7.1E-2	7.1E-2	3	
Interleukin-12 signaling	0	84	1	801	1	870	0.036	7.12E-2	7.12E-2	1	
Signaling by VEGF	0	140	1	789	1	871	0.036	7.12E-2	7.12E-2	2	
Amyloid fiber formation	0	89	1	825	1	884	0.037	7.23E-2	7.23E-2	1	
Leishmania infection	0	301	1	651	1	905	0.038	7.4E-2	7.4E-2	2	
Parasitic Infection Pathways	0	301	1	651	1	905	0.038	7.4E-2	7.4E-2	2	
Asparagine N-linked glycosylation	0	433	1	524	2	915	0.038	7.48E-2	7.48E-2	7	
G2/M Transition	0	199	1	767	1	918	0.038	7.5E-2	7.5E-2	1	
Mitotic G2-G2/M phases	0	201	1	767	1	920	0.038	7.52E-2	7.52E-2	1	
Transcriptional activity of SMAD2/SMAD3:SMAD4 heterotrimer	0	64	1	880	1	922	0.038	7.53E-2	7.53E-2	2	
Apoptotic execution phase	0	54	1	896	1	924	0.038	7.55E-2	7.55E-2	3	
Late SARS-CoV-2 Infection Events	1	98	1	875	1	939	0.039	7.67E-2	7.67E-2	21	

The corresponding pathways associated with the respective gene



Protocol references

dx.doi.org/10.17504/protocols.io.e6nvw1p9wlmk/v1