BRUNO BELLO PEDE CASTRO

Develop a pipeline to call SNPs from whole genome sequences of *Toxoplasma gondii* and integrate with conventional genotyping methods

São Paulo
2019

BRUNO BELLO PEDE CASTRO

Develop a pipeline to call SNPs from whole genome sequences of *Toxoplasma gondii* and integrate with conventional genotyping methods

Thesis submitted to the Postgraduate Program in Experimental Epidemiology Applied to Zoonosis of the School of Veterinary Medicine and Animal Science of the University of São Paulo to obtain the Doctor's degree in Sciences.

Department:
Preventive Medicine and Animal Health

Area:
Experimental Epidemiology Applied to Zoonosis

Advisor:
Prof. Solange Maria Gennari, Ph.D.

According: _____
Advisor

São Paulo
2019

Note: The original version is available at the FMVZ/USP library.

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO

(Biblioteca Virginie Buff D'Ápice da Faculdade de Medicina Veterinária e Zootecnia da Universidade de São Paulo)

T. 3805
FMVZ

Castro, Bruno Bello Pede
Develop a pipeline to call SNPs from whole genome sequences of *Toxoplasma gondii* and integrate with conventional genotyping methods / Bruno Bello Pede Castro. – 2019.
135 f. : il. + 1 CD-ROM com tabela complementar.

Título traduzido: Desenvolvimento de uma pipeline para identificação de SNPs a partir de sequências genômicas completas de *Toxoplasma gondii* e sua integração aos métodos convencionais de genotipagem.

Tese (Doutorado) – Universidade de São Paulo. Faculdade de Medicina Veterinária e Zootecnia. Departamento de Medicina Veterinária Preventiva e Saúde Animal, São Paulo, 2019.

Programa de Pós-Graduação: Epidemiologia Experimental Aplicada às Zoonoses.
Área de concentração: Epidemiologia Experimental Aplicada às Zoonoses.
Orientadora: Profa. Dra. Solange Maria Gennari.

1. Bioinformática. 2. Sequenciamento completo do genoma. 3. *Pipeline*. 4. Máquina virtual. 5. *Toxoplasma gondii*. I. Título.

Ficha catalográfica elaborada pela bibliotecária Maria Aparecida Laet, CRB 5673-8, da FMVZ/USP.

São Paulo, 21 de julho de 2015

CERTIFICADO

Certificamos que o Projeto intitulado "Desenvolvimento de uma nova geração de marcadores genéticos para estudos de epidemiologia molecular e genética de populações de Toxoplasma gondii", protocolado sob o CEUA nº 2298100215, sob a responsabilidade de Dra. Solange Maria Gennari, foi aprovado na reunião de 29/04/2015, e está de acordo com os princípios éticos da Comissão de Ética no Uso de Animais da Faculdade de Medicina Veterinária e Zootecnia da Universidade de São Paulo.

We certify that the Research "Develop a new generation of genetic markers for molecular epidemiological and population genetic studies in Toxoplasma gondii", protocol number CEUA 2298100215, under the responsibility Dra. Solange Maria Gennari, was approved in the meeting of day 04/29/2015, and agree with Ethical Principles adopted by Ethic Committee on Animal Use of the School of Veterinary Medicine and Animal Science of the University of São Paulo.

Atenciosamente,

Profa. Dra. Denise Tabacchi Fantoni
Presidente da Comissão de Ética no Uso de Animais
Faculdade de Medicina Veterinária e Zootecnia
Universidade de São Paulo

EVALUATION FORM

Author: CASTRO, Bruno Bello Pede

Title: Develop a pipeline to call SNPs from whole genome sequences of *Toxoplasma gondii* and integrate with conventional genotyping methods

Thesis submitted to the Postgraduate Program in Experimental Epidemiology Applied to Zoonosis of the School of Veterinary Medicine and Animal Science of the University of São Paulo to obtain the Doctor's degree in Sciences.

Date: _____/_____/_____

Committee members

Prof.:_____

Institution: _____ Decision: _____

Prof.:_____

Institution: _____ Decision: _____

Prof.:_____

Institution: _____ Decision: _____

Prof.:_____

Institution: _____ Decision: _____

Prof.:_____

Institution: _____ Decision: _____

DEDICATION

To Lord, for always being there for me!

ACKNOWLEDGEMENTS

I would like to thank all the people who contributed in some way to the work described in this thesis.

I would like to sincerely thank my thesis advisor, Solange Maria Gennari, for her support throughout this study and specially for her confidence in me.

To my girlfriend Flávia, thank you for your understanding and encouragement in many, many moments of crisis.

To Hernan Lorenzi, I don't know where I would be now if it wasn't for your huge help and patience in my many mistakes. You are truly a special person. I thank you from the bottom of my heart.

I am very fortunate and grateful to Dr. Chunlei Su for all the support provides me, from the food he gently took me to the hotel on my arrival at Knoxville, his patience and ideas; paramount to the realization of this thesis. Thank you for your support and helpful suggestions, you modified my life as a student and as a person. I will be forever thankful to you.

I would like to thank the Department of Preventive Medicine and Animal Health (VPS), São Paulo Research Foundation (FAPESP) for provide my scholarship in Brazil (grant 2015/23472-0) and United States (grant 2016/02646-3).

RESUMO

CASTRO, Bruno Bello Pede. Desenvolvimento de uma pipeline para identificação de SNPs a partir de sequências genômicas completas de *Toxoplasma gondii* e sua integração aos métodos convencionais de genotipagem. 2019. 120 p. Tese (Doutorado em Ciências) - Faculdade de Medicina Veterinária e Zootecnia, Universidade de São Paulo, São Paulo, 2019.

A toxoplasmose é uma doença parasitária causada pelo *Toxoplasma gondii.* O *Toxoplasma gondii* é um parasita intracelular relacionado ao *Plasmodium falciparum*, o agente causador da malária em humanos. O *Toxoplasma gondii* pode infectar todos os vertebrados homeotérmicos, incluindo mamíferos e pássaros. Recentes avanços nas tecnologias de sequenciamento de DNA tornaram possível a obtenção de sequências genômicas completas para praticamente qualquer organismo, incluindo o *Toxoplasma gondii* e com isso, a tendência é que genotipagens do tipo PCR-RFLP e MLST, atualmente utilizadas, devam ser substituídas. Uma vez que esse inestimável banco de dados gerado ao longo das últimas décadas não pode ser relacionado a essa nova tecnologia, esse trabalho teve como objetivo aliviar esse problema, desenvolvendo uma pipeline, capaz de mapear leituras provenientes de um sequenciamento genômico completo, em plataforma Illumina e identificar SNPs (*Single Nucleotide Polymorphisms*). Nesse trabalho, foram utilizados dados de sequenciamento de um total de 62 isolados de *T. gondii* provenientes de vários locais do mundo. A partir dessas sequências, foram gerados dados aprimorados para análise filogenética utilizando o software SplitsTree4 e dados de genética de populações, através da ferramenta FastStructure. Além disso, outras ferramentas que funcionam em conjunto à pipeline foram desenvolvidas, possibilitando também extrair sequências genômicas para os 10 marcadores PCR-RFLP e oito íntrons, que foram utilizados para análise genética de *T. gondii* na literatura. Para disponibilizar essas ferramentas para a comunidade de pesquisa, integramos todos os softwares e o conjunto de instruções utilizadas em

linguagem Perl, em uma máquina virtual, tornando possível a execução de tarefas de Bioinformática a partir de qualquer computador pessoal, independente do sistema operacional que estiver executando. Para isso, utilizamos um software de virtualização multiplataforma, o VirtualBox.

Palavras-chave: Bioinformática. Pipeline. Sequenciamento genômico completo. *Toxoplasma gondii*. Máquina virtual.

ABSTRACT

CASTRO, Bruno Bello Pede. Develop a pipeline to call SNPs from whole genome sequences of *Toxoplasma gondii* and integrate with conventional genotyping methods. 2019. 120 p. Tese (Doutorado em Ciências) - Faculdade de Medicina Veterinária e Zootecnia, Universidade de São Paulo, São Paulo, 2019.

Toxoplasmosis is a parasitic disease caused by *Toxoplasma gondii*. *Toxoplasma gondii* is an intracellular parasite that is related to *Plasmodium falciparum*, the agent that causes malaria in human. *Toxoplasma gondii* infects all warm-blooded vertebrates, including mammals and birds. Recent advances in DNA sequencing technologies have made it possible to obtain and use whole genome sequences to genotype any organism, including *T. gondii*. In the past, PCR-RFLP and MLST are the most common methods to genotype and identify *T. gondii* and invaluable database is generated over the last decades using these methods. However, the conventional PCR-RFLP and MLST data cannot be easily integrated with the whole genome sequence typing. The objective of this work is to develop a pipeline to map reads coming from a whole genome sequencing to identify SNPs (Single Nucleotide Polymorphisms), and to integrate the data with PCR-RFLP and MLST data. In this work, we used sequencing data from a total of 62 *T. gondii* isolates from various locations around the world. From these sequences, improved data for phylogenetic analysis were generated using the SplitsTree4 software and population genetics data through the FastStructure tool. In addition, other tools that work in conjunction with the pipeline were developed, making it possible to extract genomic sequences for the 10 PCR-RFLP markers and eight introns for MLST, which were used for genetic analysis of *T. gondii* in the literature. To make these tools available to the research community; we integrate all software and instruction set used in Perl scripts into a virtual machine, making it possible to perform Bioinformatics tasks from any personal computer, regardless of the operating system running. For this,

we use multiplatform virtualization software, VirtualBox. Implement of these tools will facility molecular genetics and population genetics of *T. gondii*. These tools can be easily modified to work with other organisms as needed.

Keywords: Bioinformatics. Pipeline. Whole genomic sequencing. *Toxoplasma gondii*. Virtual machine.

SUMMARY

1. INTRODUCTION

Toxoplasmosis is a parasitic disease caused by *Toxoplasma gondii*. An intracellular parasite that is related to *Plasmodium falciparum*, the agent that causes malaria in human. Similar to many other members of the Apicomplexa phylum, *T. gondii* shows a complex life cycle, divided between sexual and asexual phases, which are correlated with feline and non-feline infections, respectively. The sexual phase occurs only in felines (wild and domestic cats) and the asexual multiplication occurs in intermediate hosts including mammals and birds (DUBEY, 1986).

The asexual component consists of two distinct stages of growth of parasite: Tachyzoite and bradyzoite. The tachyzoite stage defines the rapidly growing form of the parasite found during the acute phase of toxoplasmosis. The tachyzoites replicate inside the host cell until the cell is lysed to release the parasites which continue to infect neighboring cells. Due host immune responses to acute toxoplasmosis infection, the tachyzoites differentiate into bradyzoites and form tissue cysts where they may reside for the life of the host.

The development of tissue cysts predominantly in the central nervous system and muscle tissue defines the chronic infection. Normally, the immune response efficiently controls the infection and prevent clinical disease. However, in immunocompromised hosts, reactivation of the parasites may occur, which can be potentially fatal.

In sexual phase, the oocysts formation occurs in felines intestine, the definitive hosts of the parasite, which are released along with the feces, contaminating the environment. In cats, the asexual multiplication (tachyzoites and bradyzoites) also occur during chronical infection.

The host can become infected by any of the three forms of infection. For humans, the infection happens by ingesting of tissue cysts from undercooked meat, consume of

food or drink contaminated with oocysts or by vertical transmission from mother to fetus (HILL; DUBEY, 2002).

Scientists in Brazil have made important contribution to the study of *T. gondii* in history. *Toxoplasma gondii* was first described in 1908 by Brazilian scientist Alfonso Splendore in São Paulo and French scientists Charles Nicolle and Louis Manceaux at the Institute Pasteur in Tunis.

T. gondii can cause abnormal development of fetus in pregnant women, blindness and eye damage in healthy individuals. In immunocompromised individuals such as AIDS patients and recipients of organ transplant, *T. gondii* reactivates and replicates rapidly, which may cause severe encephalitis and lead to high mortality in patients (MONTOYA; LIESENFELD, 2004) and *T. gondii* strains are highly diverse and some of them seemed to be associated with severe toxoplasmosis in healthy adults (BOSSI; BRICAIRE, 2004; MOURA et al., 2006). These parasite strain-specific diseases are of importance for future studies of *T. gondii* pathogenesis. However, the epidemiology and diversity of *T. gondii* is not well understood.

Understanding *T. gondii* population structure is of great interest, as it may provide us with essential information regarding the transmission and evolution of this widespread zoonotic parasite.

Isolates of *T. gondii* from South America, specially from Brazil were biologically and genetically different from those in North America and Europe. While *T. gondii* population in Brazil has an epidemic population structure with a highly genetic diversity, with a few successful clonal lineages (PENA et al., 2008), the population structure in North America and Europe is clonal, in which three clonal lineages (types I, II and III) is predominant (HOWE; SIBLEY, 1995).

Archetypal type II and type III are dominant in Europe and North Africa, while types II, III and 12 are dominant in North America, and Chinese type 1 is most

prevalent in East Asia. In contrast, *T. gondii* strains in South America are highly diverse with no clear dominance of any particular genotypes (SHWAB et al., 2014).

The majority of genotypes prevalent in Europe, North America, North Africa and Asia are non-lethal to mice at low infection dose, whereas a large proportion of *T. gondii* strains identified in South America are highly virulent and lethal to mice.

The low genetic diversity of *T. gondii* observed in the northern hemisphere may be a result of the loss of genetic variation that occurs when a new population is established by a very small number of individuals from a larger population (founder effect). The high genetic diversity observed in South and Central America suggests that *T. gondii* originated in this region (LEHMANN et al., 2006).

T. gondii is the second most common cause of foodborne illness in the United States (BATZ; HOFFMANN; MORRIS JR., 2011). On average, one-third of the human population worldwide is chronically infected with this parasite (MONTOYA; LIESENFELD, 2004; DUBEY, 2010). In Brazil, the infection rate in women of child-bearing age is greater than 60% (PAPPAS; ROUSSOS; FALAGAS, 2009).

A recent review of literature showed that, Brazil has a very high rate of *T. gondii* infection in humans (DUBEY et al., 2012). Up to 50% of elementary school children and 80% of women of child-bearing age have antibodies to this parasite. The burden of toxoplasmosis in congenitally infected children is also very high. On screening of infants at birth, 5-23 children are born infected per 10000 live births. Among the congenitally infected children, 35% had neurological symptoms (hydrocephalus, microcephaly and mental retardation), and 80% had ocular lesions. The severity of clinical toxoplasmosis in Brazilian children may be associated with certain genotypes of *T. gondii* isolates prevailing in animals and humans in Brazil (DUBEY et al., 2012). In the past decade, researchers from Brazil have published numerous articles on molecular epidemiology and genetic diversity of *T. gondii*. These studies revealed that *T. gondii* infection level in Brazil is very high comparing to other regions of the world,

and toxoplasmosis is indeed an important parasitic disease affecting human health in this country.

Given that *T. gondii* strains in Brazil are highly diverse and some genotypes may be highly virulent to humans, it is necessary to develop genetic tools to distinguish different parasite strains. Currently, a set of 10 PCR-RFLP markers have been widely used for such purpose and identified nearly 200 genotypes worldwide (SU; ZHANG; DUBEY, 2006; SHWAB et al., 2014). Though these markers have a high resolution in identifying different *T. gondii* strains, they cover 8 chromosomes of the parasite and may miss additional genetic information. The genome *Toxoplasma gondii* is compounded of 14 chromosomes and approximately 65 Mbp and about 50% of GC.

With the emergence and cheapening of whole genome sequencing techniques, new possibilities can be explored to understand this parasite and taking advantage of this data. The genome differences between the *T. gondii* strains can contribute towards better understanding on different virulence attributed to these lineages. In this context, the objective of this report is to develop tools that can be integrated to the techniques currently used.

Thus, the second chapter reports the development of a VirtualBox pipeline to process Illumina whole genome sequencing reads for Single Nucleotide Polymorphism (SNP) identification and analysis of *T. gondii* and an integration of all software and instruction set used in Perl scripts into a virtual machine.

The third chapter describes the development of a pipeline to assemble PCR-RFLP markers sequences from whole genome sequences to infer multilocus polymerase chain reaction-restriction fragment length polymorphism (PCR-RFLP) and multilocus sequence typing (MLST) genotypes. The formatting of chapters two and three follows the structure of journal article, as both are in progress for publication.

1.1. NAMES AND ADDRESS OF THE COLLABORATORS

Dr. Hernan Lorenzi, Department of Infectious Diseases, the J. Craig Venter Institute, Rockville, MD, USA. Responsible for whole genome sequencing analysis and identify SNPs.

Dr. Chunlei Su. Department of Microbiology, the University of Tennessee, Knoxville, TN, USA. Responsible for project design and coordinate the study.

2. DEVELOP A VIRTUALBOX PIPELINE TO PROCESS ILLUMINA WHOLE GENOME SEQUENCING READS FOR SNP ANALYSIS OF *T. gondii*

ABSTRACT

Recent advances in DNA sequencing technologies have made it possible to obtain whole genomic sequences for virtually any organism, including *Toxoplasma gondii* and with that, the purpose of this report is develop a pipeline to process Illumina whole genome sequences to identify Single Nucleotide Polymorphisms (SNPs) for phylogenetic and genetic structure analysis of the parasite. The pipeline is composed of a series of programs, where, reads obtained from an NCBI database are treated, mapped to reference *T. gondii* reference genome ME49 and subjected to some processes that involve sorting, duplicating, indexing, and realigning the reads in insertion or deletion (InDels) regions, so that the end of this step, a BAM extension file is obtained, which allows the identification of the SNPs. All process was performed using commands via terminal on a computer with Linux operating system. A number of software that are used for bioinformatics analysis of whole genome sequences were integrated into a package that can be used in Oracle VirtualBox, in order to these tools available to the research community.

Keywords: Pipeline. VirtualBox. Whole genome sequences. *Toxoplasma gondii*. Single Nucleotide Polymorphisms.

2.1. INTRODUCTION

The purpose of the pipeline is to process Illumina whole genome sequences and map the reads to reference *T. gondii* reference genome ME49 strain, identify SNPs for phylogenetic and genetic structure analysis of the parasite.

We aimed to have the following functions of the pipeline: to map Illumina sequence reads to ME49 and identify SNPs, generate appropriate data files for phylogenetic analysis by SplitsTree4, and population genetic analysis by FastStructure. In addition, we also aimed to extract sequence data for the 10 PCR-RFLP markers and 8 introns (MLST) that has been used for genetic analysis of *T. gondii* in literature.

This first stage of the project was to develop pipeline to identify SNP for a total of 62 *T. gondii* isolates from different locations around the world. These SNPs contain important information that can be analyzed using Bioinformatics tools. The entire process was performed using commands via terminal on a computer with Linux operating system. A number of software that are used for bioinformatics analysis of whole genome sequences were integrated into a package that can be used in Oracle VirtualBox. A general flowchart of this process is outlined below (Figure 1).

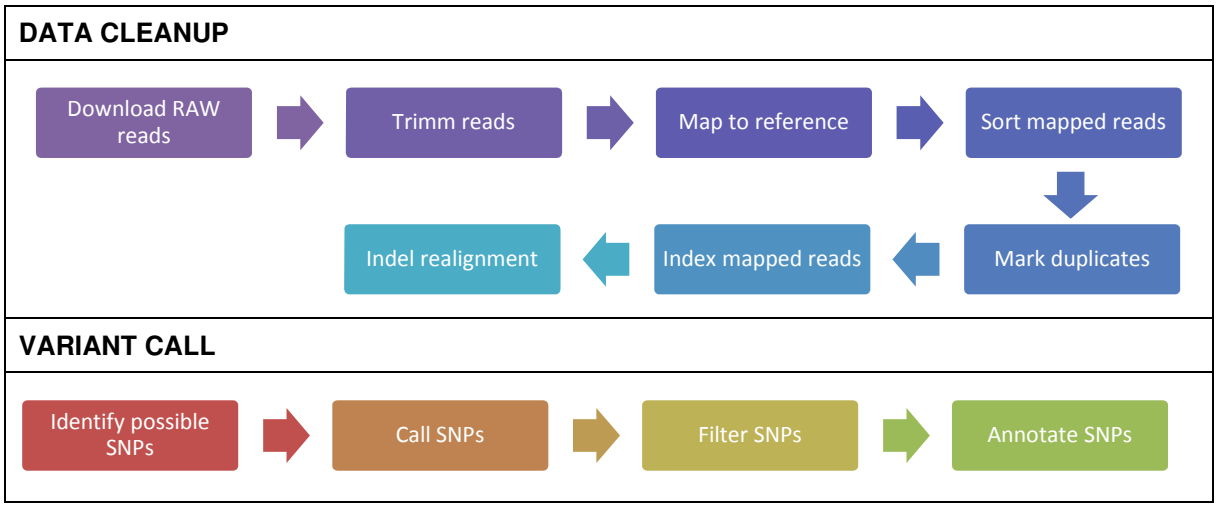


Figure 1: A flowchart of whole genome analysis to identify SNPs for *T. gondii*.

2.2. MATERIALS AND METHODS

2.2.1. Download whole genome sequences

The first step was to begin downloading the genomic sequences already available on a specific NCBI database, known as **SRA** (Sequence Read Archive), which stores and provides raw sequencing data and alignment information for new sequencing methodologies. For this, we used the **fastq-dump**, which is part of the SRA Toolkit package, version 2.8.2, available at: http://www.ncbi.nlm.nih.gov/sra/docs/toolkitsoft (LEINONEN et al., 2010). This tool allows sequencing data to be retrieved from the SRA repository and converted to FASTQ format, through an accession number assigned to each genomic sequence submitted to the NCBI repository. The complete syntax for using this utility is as follows:

```
fastq-dump --split-3 --gzip -A SRR000000
```

The **--split-3** parameter indicates that each read must be downloaded in a separate file, considering that the sequences used in this project are paired-end, we have two read files for each sequence. The first direct read sequence is given the suffix R1, while the second sequence, the reverse sequence is given the suffix R2. The **--gzip** parameter compresses the files, putting them in the smallest gzip format. The **-A** parameter indicates the accession number to be downloaded.

2.2.2. Map short sequence reads to reference genome ME49

To identify SNPs, the raw sequences will be processed, mapped to reference genome, and identify the variations. The flow chart below shows the main process:

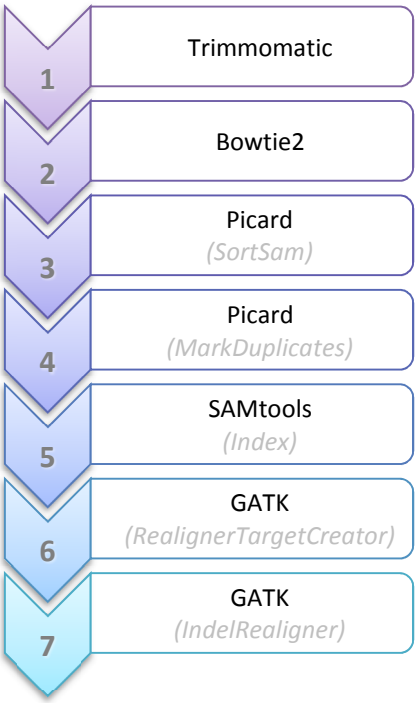


Figure 2: A flowchart of the pipeline process.

2.2.2.1. Trim Illumina short sequence reads

The pipeline starts with the complete FASTQ genomic sequences submitted to Trimmomatic software, version 0.36, available for download from the website: http://www.usadellab.org/cms/index.php?page=trimmomatic (BOLGER; LOHSE; USADEL, 2014). This tool removes adapters and bases with lower-than-expected read quality in genomic sequences, minimizing errors generated during sequencing. Having "cleaner" or "trimmed" reads decreases pipeline runtime, as well as anticipates the elimination of reads that would be discarded in the alignment step by having a high number of mismatches against the genome. The pipeline uses the following command to run Trimmomatic:

```
java -jar Trimmomatic.jar PE -threads 10 -phred33 $strain.R1.fastq.gz $strain.R2.fastq.gz
$strain.paired.R1.fastq.gz $strain.unpaired.R1.fastq.gz $strain.paired.R2.fastq.gz
$strain.unpaired.R2.fastq.gz ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 LEADING:3 TRAILING:3
SLIDINGWINDOW:4:15 MINLEN:50
```

The **java -jar** syntax indicates that a jar extension, **Trimmomatic.jar**, is running, configured to process paired-end reads (initials **PE**). The **-threads** parameter indicates the number of tasks that can be executed simultaneously, thereby reducing process execution time. The **-phred33** parameter specifies the encoding used to represent the base quality of the input FASTQ files. The paired-end mode indicates the use of two input files, with the forward and reverse reads, represented by the variables **$strain.R1.fastq.gz** and **$strain.R2.fastq.gz** respectively. A step of identifying and removing the Illumina adapters is represented by the expression **ILLUMINACLIP:TruSeq3-PE.fa:2:30:10**. The parameter **LEADING:3** removes bases from the beginning with a quality less than 3 or with the number of bases smaller than 3. **TRAILING:3** works similarly to the previous one, but to remove bases from the end of the sequence. The **SLIDINGWINDOW:4:15** parameter indicates that the Trimmomatic must scroll through reads with a window size 4, removing when the average quality per base is less than 15. The last parameter, **MINLEN:50**, suggests that reads with lower compliance than 50 bases. The pipeline automatically runs Trimmomatic for all sequences listed in the List file, generating a total of four output files for each strain, being: two paired files, (**$strain.paired.R1.fastq.gz** and **$strain.paired.R2.fastq.gz**) and two unpaired (**$strain.unpaired.R1.fastq.gz** and **$strain.unpaired.R2.fastq.gz**). At the end of the Trimmomatic process, the pipeline submits these filtered files for alignment to reference genome ME49.

2.2.2.2. Map processed Illumina sequence to reference genome ME49

Map Illumina sequences to reference genome is achieved by software **Bowtie2**, which performs the alignment of these reads against long reference sequences, in which case we use the ME49 strain genome. The version of Bowtie2 used in this pipeline was 2.3.4.1 (LANGMEAD; SALZBERG, 2012), downloaded through the address: http://bowtie-bio.sourceforge.net/bowtie2/index.shtml. The command for Bowtie2 is:

```
bowtie2 --rg-id $strain --rg PL:Illumina --rg SM:$strain --rg LB:$strain -p 10 --phred33 -x
ToxoDB-13.0_TgondiiME49 -1 $strain.paired.R1.fastq.gz -2 $strain.paired.R2.fastq.gz -U
$strain.unpaired.R1.fastq.gz,$strain.unpaired.R2.fastq.gz > $strain.bam
```

The **bowtie2** software runs and the read group ID is set with the prefix name (parameter **--rg-id**). Using the **--rg** parameter allows adding information to the header of the BAM output file (usually of the form `TAG:VALUE`). The **-p** parameter informs the number of server/computer cores that will be used by the program for data processing (the more cores, the faster the processing speed). Similar to Trimmomatic, the use of the **--phred33** parameter indicates that the base quality representation uses Phred33 encoding. The **-x** parameter suggests the index file to be used; in this case, we used the reference genome for the ME49 isolate. The paired reads, direct and reverse respectively, are passed through the parameters **-1** and **-2** and the **-U** parameter indicates the unpaired sequences that will be aligned. Using the character > indicates that the terminal should redirect the information to an output file, which will contain the aligned reads and BAM extension.

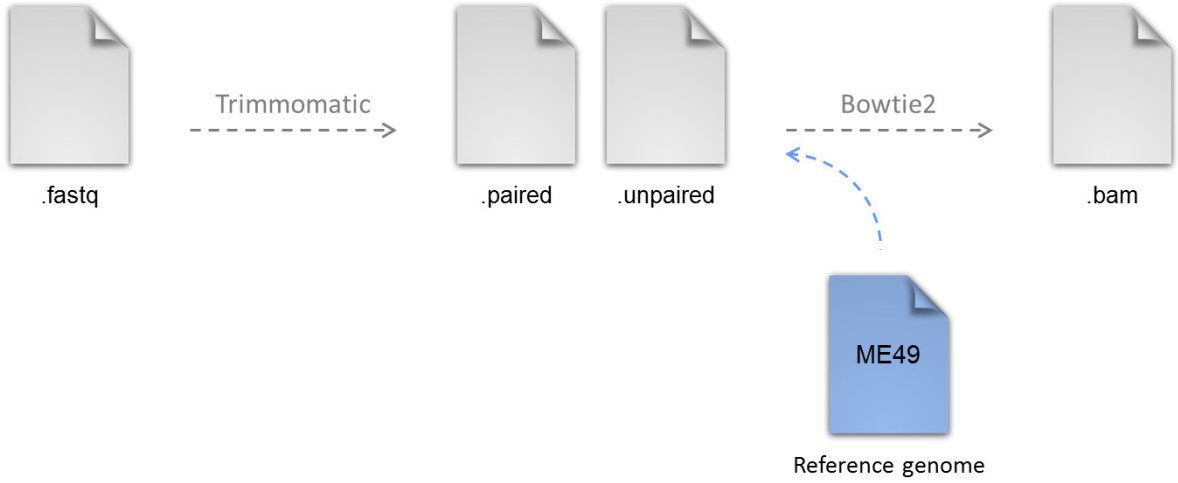


Figure 3: Representation of the input and output files for the use of the Trimmomatic and Bowtie2 software.

Sort mapped sequences

The output file of the Bowtie2 software, BAM file, is used by the pipeline as an input file for the **SortSam** program (version 2.6.0), a software developed in Java and belonging a **Picard** package (http://broadinstitute.github.io/picard), that manipulates SAM/BAM files (**Picard Toolkit**, 2019). The syntax used by the pipeline to run SortSam has the following:

```
java -Xmx2g -jar Picard.jar SortSam INPUT=$strain.bam OUTPUT=$strain.sorted.bam SO=coordinate
TMP_DIR=Temp
```

In this case, the input file indicated by the **INPUT** variable is sorted according to the argument indicated in the **SO** (Sort Order) parameter. The "coordinate" value tells SortSam must sort the reads contained in the input file first by the chromosome and then by the starting position. The output file, informed by the variable **OUTPUT** and SORTED.BAM extension will contain, basically, the same information as the input BAM file, but ordered.

2.2.2.3. Mark duplicate sequences

The next step locates and marks duplicate reads of the files ordered in the previous step by using another software from Picard package (**Picard Toolkit**, 2019), the **MarkDuplicates**. During the sequencing process, duplication of a fragment may occur and it is recommended that suspected duplicate reads be marked and ignored in subsequent analyzes. MarkDuplicates looks for reads where the 5' alignment and orientation positions are the same, and compares them through an algorithm, so that the read with the best quality (primary read) is not changed and the others are marked as duplicates. The command used to run the MarkDuplicates function is:

```
java -jar Picard.jar MarkDuplicates INPUT=$strain.sorted.bam OUTPUT=$strain.sorted.dedup.bam
METRICS_FILE=$strain.sorted.metrics
```

The input file is indicated by the **INPUT** parameter and the output is defined in two arguments, the first one being assigned to the **OUTPUT** parameter, where a deduplicated file with SORTED.DEDUP.BAM extension is generated, and a second, the parameter **METRICS_FILE**, whose product is a METRICS extension file, that contains a set of metrics describing the duplicates found in the input BAM file.

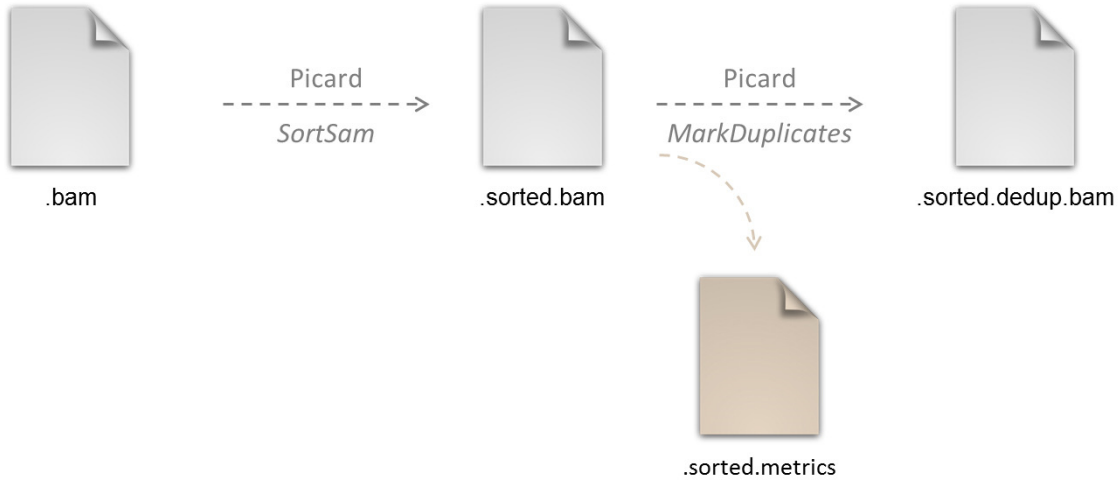


Figure 4: Representation of input and output files for SortSam and MarkDuplicates tools (Picard).

2.2.2.4. Index the mapped and deduplicated sequences

Many software requires an indexed BAM file in order to access it more efficiently. Indexing is performed by the **index** function, from **SAMtools** software (LI et al., 2009), version 1.7, available for download through the website: http://samtools.sourceforge.net/. The following pattern of command generates a corresponding index file, with BAI extension.

```
samtools index $strain.sorted.dedup.bam
```

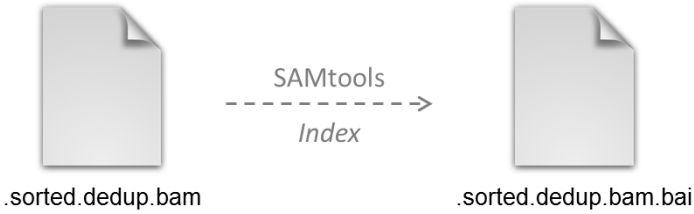


Figure 5: Representation of input and output files for use of the SAMtools index function.

2.2.2.4. Realign mapped sequences (RealignerTargetCreator)

Because software that perform alignment are optimized to align thousands of reads quickly, errors are common, especially where insertions or deletions occur (InDels). In order to minimize this bias, a two-stage local realignment is performed.

In the first step, the **RealignerTargetCreator** function from the **Genome Analysis Toolkit** software, **GATK** (MCKENNA et al., 2010), version 3.6-0, available at: http://www.broadinstitute.org/gatk is used. This tool identifies small suspicious intervals, which may require realignment. The **-T** parameter described in the command indicates the function used, RealignerTargetCreator. The **-R** parameter indicates the reference genome of the ME49 strain used in the alignment. The command also suggests the input file, indicated by the **-I** argument and the output file, which contains a list with the realigned ranges and INTERVALS.LIST extension, reported by the **-o** parameter.

```
java -jar GenomeAnalysisTK.jar -T RealignerTargetCreator -R ToxoDB-13.0_TgondiiME49.fa -I $strain.sorted.dedup.bam -o $strain.target.intervals.list
```

2.2.2.5. Realign mapped sequences (IndelRealigner)

Upon completion, the pipeline performs another function from the GATK package, the **IndelRealigner**, which performs better quality realignment based on the

INTERVALS.LIST file generated in the previous step. The command format is very similar to the previous one, including only the **-targetIntervals** parameter that contains the list with the intervals.

```
java -Xmx4g -jar GenomeAnalysisTK.jar -T IndelRealigner -R ToxoDB-13.0_TgondiiME49.fa -I
$strain.sorted.dedup.bam -targetIntervals $strain.target.intervals.list -o
$strain.sorted.dedup.realigned.reads.bam
```

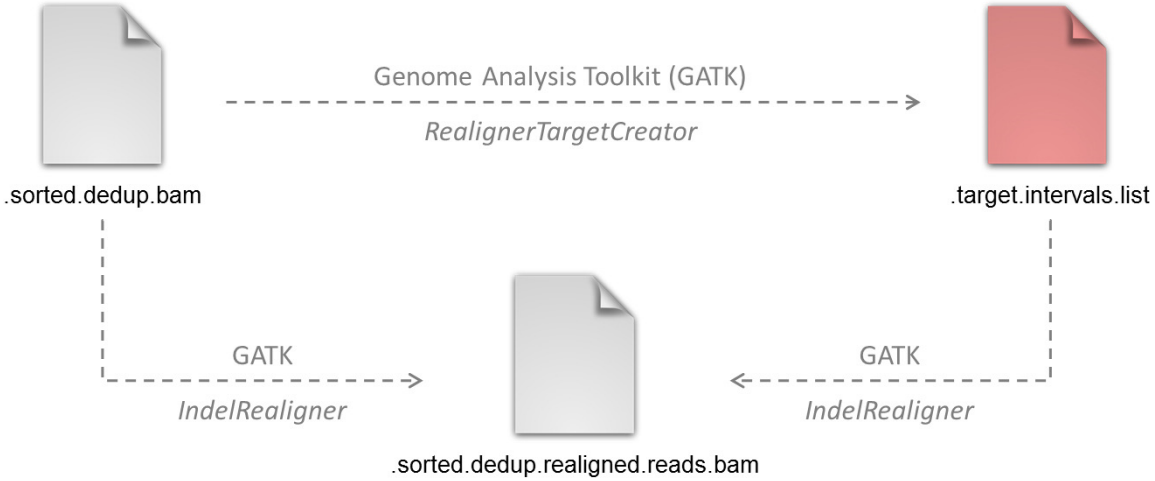


Figure 6: Representation of input and output files for use of the GATK RealignerTargetCreator and IndelRealigner functions.

2.2.3. Call SNPs

In order to optimize time, the pipelines were executed in parallel and in groups with 10 isolates, requiring a time of approximately 20 uninterrupted days for the generation of all SORTED.DEDUP.REALIGNED.READS.BAM files for the 62 *T. gondii* isolates used in this project. If the pipeline parameter **-nosnp** was set as True, at the time of the pipeline execution, the process would finish after completing all the steps described above and the realigned reads in BAM files would be further processed for SNP calling. However, we used a -nosnp parameter as F (False) and executed all below commands manually (section 2.2.4), the option comes from the fact that we

would like to analyze the SNPs separately by chromosomes and the pipeline does not have such a feature. Below is the overall flow of the processes (Figure 7):

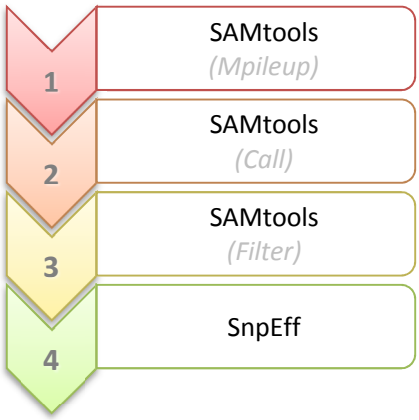


Figure 7: A flowchart of the pipeline process.

2.2.3.1. Identify variants from reference genome

The BAM files were then submitted to the **mpileup** function from **SAMtools** software (LI et al., 2009), version 1.7, which map the reads of the BAM files against a reference sequence and locate the positions where the mapped reads contain variants relative to the reference genome. The mpileup provides a summary of the coverage of mapped reads in a reference sequence in a resolution of a single pair of bases. The generic command for the function is:

```
samtools mpileup -o $strain.mpileup.bcf -t DP,DPR,DV,DP4,INFO/DPR,SP -g -s -u -f ToxoDB-13.0_TgondiiME49.fa $strain1.sorted.dedup.realigned.reads.bam $strain2.sorted.dedup.realigned.reads.bam $strain3.sorted.dedup.realigned.reads.bam [...]
```

The **samtools** command invokes the execution of the SAMtools software, through the **mpileup** function. The output file, defined by the **-o** parameter, indicates a generation of a BCF file containing the positions that have variants. The **-t** parameter points to a comma-separated list of FORMAT and INFO to output: **DP** (Number of

high-quality bases, FORMAT), **DPR** (Number of high-quality bases for each observed allele, FORMAT), **DV** (Number of high-quality non-reference bases, FORMAT), **DP4** (Number of high-quality ref-forward, ref-reverse, alt-forward and alt-reverse bases, FORMAT), **INFO/DPR** (Number of high-quality bases for each observed allele, INFO) and **SP** (Phred-scaled strand bias P-value, FORMAT). The parameter **-g** computes genotype likehoods and generates an output VCF file. **-s** indicates the output mapping quality and **-u** points a generation of an uncompressed output file. The faidx-indexed reference genome file in the FASTA format is indicated in parameter **-f**. The user must also indicate all BAM file(s) in which the variants will be identified, using spaces to separate them.

2.2.3.2. Call SNPs from all sequences

The BCF file, generated in previous mpileup step, is used as input for the **call** function, **BCFtools** software. Currently we use version 1.7 of this program (LI et al., 2009), available at: http://samtools.github.io/bcftools/bcftools.html. The call function interprets the inconsistencies between alignment reads and the reference genome sequences as variants and performs the conversion of the binary representation of the file (BCF) to text (VCF). Below is the syntax of the command line:

```
bcftools call -cv -p 0.05 $strain.mpileup.bcf > $strain.mpileup.vcf
```

The **-c** parameter represents the consensus-caller using (the original calling method) and **-v** indicates output variants sites only. The **-p** parameter is a pval-threshold and means the variant is accepted if P(ref|D) $<$ 0.05. The generation of output VCF file is indicated by the use of the character $>$.

2.2.3.3. Filter SNPs from VCF file

Since not all sequence variants identified (saved in the VCF file) are of good quality, it is necessary to remove low quality SNPs using the **filter** function, **BCFtools** (LI et al., 2009). Several criteria are used in the filter function, such as the quality of base calling, the alignment, and range bias or position bias. The command that filters the variants contained in the VCF file generates a file with extension FILTER.VCF. Below is the syntax of the command line:

```
bcftools filter -sLowQual -g3 -G10 -e'%QUAL<10 || (%MAX(DV)<=3 && FMT/GT="./1")|| (%MAX(DV)/%MAX(DP)<=0.3 && FMT/GT="./1") || %MAX(FMT/DP)<5 || (RPB>=0 && RPB<0.1 && %QUAL<15) || (MQB>=0 && MQB<0.05) || (BQB>=0 && BQB<0.05) || (MQSB>=0 && MQSB<0.0)' $strain.mpileup.vcf > $strain.mpileup.filter.vcf
```

2.2.3.4. Annotate SNPs from VCF file

After completion of the filtration step, the type of SNP was detected, classified and annotated, using the SnpEff software (CINGOLANI et al., 2012), commonly used to perform variant annotation and prediction of effect. The version 4.2 was used and is available for download at: http://snpeff.sourceforge.net/. The command to use the SnpEff is:

```
java -Xmx4g -jar snpEff.jar -ud 1000 ToxoDB-13.0_TgondiiME49 $strain.mpileup.filter.vcf > $strain.mpileup.filter.annotation.vcf
```

This program not only classifies the type of SNP but also predicts and reports amino acid changes at the protein level. The parameter **java** indicates a use of a Java program, the **snpEff.jar**. The parameter **-ud** set the length of upstream downstream interval, in this case, 1000 bases.

2.2.4. Alternative approach - Split data by chromosomes

As mentioned previously, the process of calling SNPs was performed manually, with the purpose of analyzing the SNPs separately by chromosomes. In this case, the function **for** is used in all steps, also called a loop repeat, allows to run commands in sequence for all chromosomes and the Apicoplast present in the Chromosomes file, whereas the variable **$chr** receives the name of the chromosome and a total of 14 files is generated at the end of the command, one for each chromosome. The commands for the process are:

2.2.4.1. Mpileup

```
for chr in `cat Chromosomes`
      do
      samtools mpileup -r $chr:1-10,000,000 -o $chr.mpileup.bcf -t DP,DPR,DV,DP4,INFO/DPR,SP -
      g   -s   -u   -f   ToxoDB-13.0_TgondiiME49.fa   $strain1.sorted.dedup.realigned.reads.bam
      $strain2.sorted.dedup.realigned.reads.bam $strain3.sorted.dedup.realigned.reads.bam [...]
done
```

There is a difference in this command with respect to the command used in section 0, where the **-r** parameter indicates the generation of mpileup in regions for each chromosome.

2.2.4.2. Call

```
for chr in `cat Chromosomes`
      do
      bcftools call -cv -p 0.05 $chr.mpileup.bcf > $chr.mpileup.vcf
done
```

2.2.4.3. Filter

```
for chr in `cat Chromosomes`
      do
      bcftools filter -sLowQual -g3 -G10 -e'%QUAL<10 || (%MAX(DV)<=3 && FMT/GT="./1")||
      (%MAX(DV)/%MAX(DP)<=0.3 && FMT/GT="./1") || %MAX(FMT/DP)<5 || (RPB>=0 && RPB<0.1 &&
      %QUAL<15) || (MQB>=0 && MQB<0.05) || (BQB>=0 && BQB<0.05) || (MQSB>=0 && MQSB<0.0)'
      $chr.mpileup.vcf > $chr.mpileup.filter.vcf
done
```

2.2.4.4. SnpEff

```
for chr in `cat Chromosomes`
      do
      java -Xmx4g -jar snpEff.jar -ud 1000 ToxoDB-13.0_TgondiiME49 $chr.mpileup.filter.vcf >
      $chr.mpileup.filter.annotation.vcf
done
```

These files were used and supported the creation of phylogenetic trees based on SNPs in each of the 14 chromosomes. For the creation of these phylogenetic trees, we used the software SplitsTree4 (HUSON; BRYANT, 2005). A population structure analyze of 62 *T. gondii* strains was inferred from SNP data, using the software FastStructure (RAJ; STEPHENS; PRITCHARD, 2014).

2.2.5. Develop a pipeline to assemble PCR-RFLP markers sequences from whole genome sequences to infer PCR-RFLP genotypes

Since we have available the complete genome of 62 *T. gondii* isolates, it is possible to extract the sequences corresponding to the 10 PCR-RFLP markers currently used (SU; ZHANG; DUBEY, 2006), using a tool called Pilon (WALKER et al., 2014), version 1.22 (http://www.broadinstitute.org/software/pilon). With the extracted sequences, a Perl script is able to assemble PCR-RFLP markers sequences.

2.2.6. Integration of the SNP calling processes and data analysis into pipelines

2.2.6.1. SNP call pipeline

The multiple steps needed to identify SNPs can be integrated into a pipeline for automation. executed by a Perl script. Below is the commend line:

```
call_snps_pipeline.pl -r ToxoDB-13.0_TgondiiME49.fa -nosnp T -f List -A ME49
```

The **call_snps_pipeline.pl** syntax indicates that a script is being run in the Perl programming language. The **-r** parameter indicates the reference genome, which in this case is ME49, the **-nosnp** parameter set with the variable T (True) indicates that the pipeline must abort the identification instructions of calling SNPs. The **-f** variable indicates a list with the prefixes of the samples that will be executed in the pipeline described in the file named List. The last parameter, **-A** indicates the reference annotation, which in this case is also the reference strain ME49.

As multiple steps are involved in SNP calling and data analysis, it is tedious to input command line one at a time. To address this issue, we developed Perl scripts to streamline the process. Five Perl scripts are developed to handle SNP calling and data analysis.

1) To download a list of Illumina whole genome sequences from NCBI website: `download_sequences.pl`

2) To call SNPs from Illumina whole genome sequences using a pipeline: `call_snps_pipeline.pl`

3) To extract DNA sequences of interest based on coordinates in ME49: `extract_sequences.pl`

4) To prepare SNP data for analysis using the FastStructure:
`faststructure.pl`

5) To generate multilocus PCR-RFLP profiles from SNP data:
`rflp_analysis.pl`

To make these tools available to research community, we integrated relevant bioinformatics software and the Perl scripts into a Virtual Machine. This is to make the tools portable and can be used on any computer regardless of their original operating systems. For this, we used a cross-platform virtualization application, the Oracle VM VirtualBox.

2.2.6.2. Install Oracle VM VirtualBox

We installed the VirtualBox, version 5.2.18 (**Oracle VM VirtualBox**, 2019) onto PC with Windows 10 operating system using the standard configuration.

The VirtualBox installer was downloaded from the developer's website: http://www.virtualbox.org/wiki/downloads and supports 32 and 64-bit operating systems. The program was installed using its default settings. After installing VirtualBox successfully and restart the computer, it was necessary extend the basic functionality of the VirtualBox installing an Extension Pack. The features added by the extension package are:

- Support for USB 2.0 devices (EHCI);
- Support for USB 3.0 devices (xHCI);
- Experimental support for PCI devices in Linux host;
- Intel PXE boot ROM and NVMe;
- Access the webcam from the host computer;

- Disk image encryption with AES algorithm;
- VirtualBox RDP (Remote Desktop Protocol);

The Extension Pack was downloaded from the link **All platforms** in **Oracle VM VirtualBox Extension Pack** section, available on VirtualBox download page: http://www.virtualbox.org/wiki/downloads. The installation was done through double-click on the downloaded file (*.vbox-extpack extension) and click in **Install** button. After installing and configuring VirtualBox, by installing the Extension Pack, a new Linux Bioinformatics VM has been created, the ISO Linux was required and we used a variation of Ubuntu distribution, called **Xubuntu**. The main difference between both is Xubuntu uses a XFCE desktop environment while Ubuntu uses GNOME. XFCE is a smaller version of GNOME, indicated to modest computer, that no has a high performance and this is the reason for choosing Xubuntu instead Ubuntu. The ISO Linux was downloaded from Xubuntu website: http://xubuntu.org/download and we used an 18.04.1 LTS (Long Term Support) 64-bit version.

The original Bioinformatics VM share 8 Gb of memory, 2 cores of processors and has 1 Tb of hard drive. The VM uses a dynamically file to store the information (Virtual Disk Image - VDI file), which increase in size as the information is stored. It's not possible to change the size of HD through VirtualBox menu. We used the Linux install configuration as default and this set in User's Configuration.

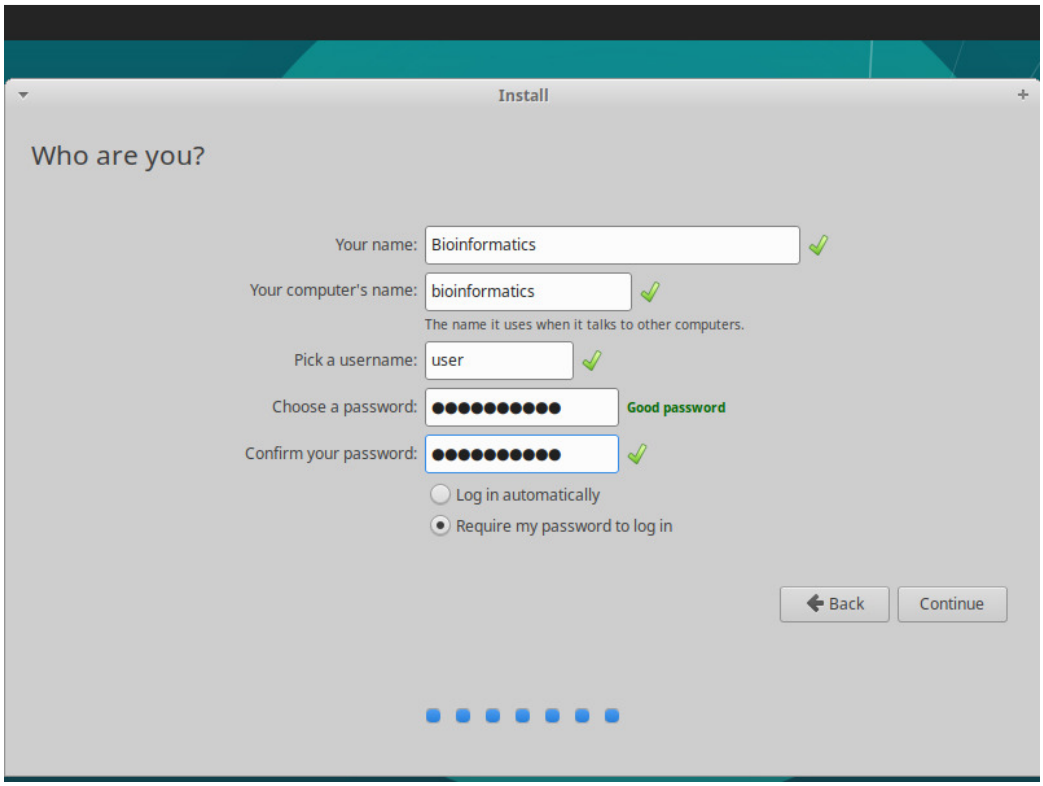


Figure 8: User Configuration used by Linux Bioinformatics VM.

The Figure 8 shows the computer's name (**bioinformatics**), a username (**user**) and the password (**bioinfo123**) used to configured the Bioinformatics VM, the user must to use the password to install different programs or change some configuration.

Before to install the programs required to run the pipeline, the VirtualBox Guest Additions must be installed, clicking in Devices Menu → Insert Guest Additions CD Image). Once CD image is mounted, the below command was used in terminal:

- **To install:**

```
cd /media/user/VBox_GAs_5.2.18
sudo ./VBoxLinuxAdditions.run
```

- **To check version:**

```
VBoxClient --version
Version: 5.2.18
```

2.2.6.3. Install and configure relevant Bioinformatics software

The tools available in Bioinformatics VM were installed in this order, using the respectively command and check version in terminal.

1) SRA Toolkit

- **To install:**

```
sudo apt-get install sra-toolkit
```

- **To check version:**

```
fastq-dump -version
Version: 2.8.2
```

2) SAMtools

- **To install:**

```
sudo apt-get install samtools
```

- **To check version:**

```
samtools --version
Version: 1.7
```

3) Java

- **To install:**

```
sudo apt-get install openjdk-8-jre
```

- **To check version:**

```
java -version
Version: 1.8.0_191
```

4) Bowtie2

- **To install:**

```
sudo apt-get install bowtie2
```

- **To check version:**

```
bowtie2 --version
Version: 2.3.4.1
```

5) BLAST

- **To install:**

```
sudo apt-get install blast2
```

- **To check version:**

```
blastn -version
Version: 2.6.0+
```

6) BioPerl

- **To install:**

```
sudo apt-get install bioperl
```

- **To check version:**

```
perl -MBio::Root::Version -e 'print $Bio::Root::Version::VERSION."\n"'
Version: 1.007002
```

7) BCFTools

- **To install:**

```
sudo apt-get install bcftools
```

- **To check version:**

```
bcftools --version
Version: 1.7
```

8) SplitsTree4

- **To install:**

  Download the Install.sh file from website:

  http://www.splitstree.org/

  sudo sh Install.sh

- **To check version:**

  SplitsTree4 --version

  Version: 4.14.6

9) Trimmomatic

- **To install:**

  Download the *.zip file from website:

  http://www.usadellab.org/cms/index.php?page=trimmomatic

  Uncompressing the *.zip file and copy the content to:

  /home/user/Bioinformatics/Programs/Trimmomatic/0.36

- **To check version:**

  java -jar Trimmomatic.jar -version

  Version: 0.36

10) GATK

- **To install:**

  Download the *.zip file from website:

  http://software.broadinstitute.org/gatk

  Uncompressing the *.zip file and copy the content to:

  /home/user/Bioinformatics/Programs/GATK/3.6-0

- **To check version:**

  java -jar GenomeAnalysisTK.jar -version

  Version: 3.6-0

11) FastStructure

- **To install:**

  sudo apt-get install cython libgsl0-dev python-matplotlib python-numpy python-scipy

  Download the *.zip file from website:
  http://rajanil.github.io/fastStructure
  Uncompressing the zip file and copy the content to:
  /home/user/Bioinformatics/Programs/FastStructure/1.0-4

  In **1.0-4** folder, type in terminal:
  cd vars
  python setup.py build_ext --inplace
  cd ..
  python setup.py build_ext --inplace

- **To check version:**

  There is no way to check version using a terminal command. For more details about the FastStructure version, please check:
  http://rajanil.github.io/fastStructure
  Version: 1.0

12) Picard

- **To install:**

  Download the *.zip file from website:
  http://broadinstitute.github.io/picard
  Uncompressing the *.zip file and copy the content to:
  /home/user/Bioinformatics/Programs/Picard/2.6.0

- **To check version:**

  java -jar Picard.jar SortSam --version
  Version: 2.6.0

13) Pilon

- **To install:**

  Download the *.zip file from website:

  http://github.com/broadinstitute/pilon

  Uncompressing the *.zip file and copy the content to:

  /home/user/Bioinformatics/Programs/Pilon/1.22

- **To check version:**

  java -jar Pilon.jar --version

  Version: 1.22

14) SnpEff

- **To install:**

  Download the *.zip file from website:

  http://snpeff.sourceforge.net/

  Uncompressing the *.zip file and copy the content to:

  /home/user/Bioinformatics/Programs/SnpEff/4.2

- **To check version:**

  java -jar snpEff.jar -version

  Version: 4.2

The above software are integrated into the Bioinformatics VirtualBox, this package can be downloaded.

2.2.6.4. Instruction to install Bioinformatics to VB machine

1. Navigate to directory /home/user/Bioinformatics/Desktop

Files in folder Desktop are:

Coordinates of the 10 PCR-RFLP markers with IDs

```
ME49_PCR-RFLP_marker_coordinates
```

Coordinates of eight introns

```
ME49_introns_coordinates
```

A list of *T. gondii* chromosomes

```
Chr_list
```

File with list of sequences to download (empty at this moment)

```
Acc_list
```

File with list of sequences processed (empty at this moment)

```
List
```

Step-by-step procedure of using the pipeline

```
Procedures
```

2. Copy the folder "Desktop" to make new folder, rename it, such as "Working_dir".

If you plan to have multiple batches of sequences to process, copy and paste the folder "Desktop" to make a few new folders, rename them as "Working_dir1", "Working_dir2", "Working_dir3" and so on. Make folder "Working_all" to combine all processed sequences for final call of SNPs.

3. Download Illumina whole genome sequences from NCBI

To manually download a single whole genome sequence from NCBI web site (Replace the XXXXXXX in SRR number with 6 to 7 digits) in the syntax below with the sequence you will download

```
fastq-dump --split-3 --gzip -A SRRXXXXXXX
```

To download a list of sequences from NCBI web site. Prepare the list and save it in a text file named "Acc_list".

Example of "Acc_list" with two sequence SRR IDs and strain names (Tab delimited):

```
SRR3666803	TgCkBr016
SRR3666806	TgPumaMe1
```

Run perl command to download sequences:

```
download_sequences.pl -f Acc_list
```

When sequences are all downloaded from NIBI, the SRRXXXXXXX of each strain is renamed automatically to "SRRXXXXXXX_strainname" and saved in file "List" for downstream analysis.

An example of "List" file is:

```
SRR3666803_TgCkBr016
SRR3666806_TgPumaMe1
```

4. Call SNPs (complete pipeline)

The pipeline to call SNPs has two parts, the first part map Illumina short sequences to reference genome ME49, the aligned "SRRXXXXXXX_ID.sorted.dedup.realigned.reads.bam" files are generated. The second part call SNPs from the listed multiple genomes to pileup SNPs and generate a SNP summary file "All_SAMPLEs.mpileup.filter.annotation.SNP.summary.txt"

Run complete pipeline:

```
call_snps_pipeline.pl -r ToxoDB-13.0_TgondiiME49.fa -nosnp F -onlysnp F -f List -delete T -rflp T -A ME49
```

Generated: (1) SNPs and fasta file for future RFLP analysis, (2) ALL_SAMPLES.mpileup.filter.annotation.SNP.summary.txt

Intermediate bam files are deleted to save space on the disk (-delete T).

5. Generate pseudo-fasta file from ALL_SAMPLES.mpileup.filter.annotation.SNP.summary.txt

Generate ALL_SAMPLES.fasta file:

```
make_fasta_files_from_vcf_summary.pl -p ALL_SAMPLES -i ALL_SAMPLES.mpileup.filter.annotation.SNP.summary.txt -c 5 -M 0 -d
```

#- -c 5 means 5x sequence coverage.

-min 0.2 -max 0.8 mean minimum allele frequency is 20%, and maximum is 80%. If this function is not used, then keep all SNPS

-M

-d

-p ALL_SAMPLES is prefix for created files.

6. Phylogenetic analysis of SNPs by SplitsTree

From the folder Programs, launch SplitsTree, open ALL_SAMPLES.fasta to generate a phylogenetic network.

7. FastStructure data analysis

Run command line:

```
fast_structure.pl -i ALL_SAMPLES.fasta
```

This process generates the following data files for FastStructure analysis. The file "ALL_SAMPLES.str" contains all SNPs. The other files contain a subset of only 1% of SNPs. The file ALL_SAMPLES.SNP_data_01.str contains SNPs starting at position 1 and every 100th SNP forward, ALL_SAMPLES.SNP_data_20.str contains SNPs starting at position 20 and every 100th SNP forward, and so on.

```
ALL_SAMPLES.str                    # contains all SNPS
ALL_SAMPLES.SNP_data_01.str        # contains SNPS from position 1 and every 100th position
ALL_SAMPLES.SNP_data_20.str        # contains SNPS from position 20 and every 100th position
ALL_SAMPLES.SNP_data_40.str        # contains SNPS from position 40 and every 100th position
ALL_SAMPLES.SNP_data_60.str        # contains SNPS from position 60 and every 100th position
ALL_SAMPLES.SNP_data_80.str        # contains SNPS from position 80 and every 100th position
```

Generate FastStructure results

Move all FastStructure data files to: /home/user/Desktop/Bioinformatics/Programs/FastStructure/1.0-4/Data/

At the terminal type: cd /home/user/Bioinformatics/Programs/FastStructure/1.0-4/, then run the command line:

```
for a in {2..10}
        do
        echo Creating output file for K = $a...
        python     structure.py     -K     $a     --input=Data/ALL_SAMPLES.SNP_data_01     --
        output=Output/ALL_SAMPLES.SNP_data_01 --format=str --full --seed=100
done
```

#The output results are in folder: /home/user/Desktop/Bioinformatics/Programs/FastStructure/1.0-4/Output/

Choose best K value

Run command line:

```
python chooseK.py --input=Output/ALL_SAMPLES.SNP_data_01
```

Identify the best K value from the output results. For example:

Model complexity that maximizes marginal likelihood = 3

Model components used to explain structure in data = 3

Display the best K, for example: K = 3

Run command line:

```
python distruct.py -K 3 --input=Output/ALL_SAMPLES.SNP_data_01 --
output=Output/ALL_SAMPLES.SNP_data_01_k3_distruct.svg
```

8. Generate RFLP results from whole genome SNP data

Run command line:

```
rflp_analysis.pl -f List
```

Results for ToxoDB PCR-RFLP genotype are saved in file "result.rflp.analysis"

9. Extract DNA sequences for the 10 PCR-RFLP markers

Run command line:

```
extract_sequences.pl -r ToxoDB-13.0_TgondiiME49.fa -f List -c ME49_PCR-RFLP_marker_coordinates
-o PCR-RFLP_marker_sequences -A ME49
```

Results for marker sequences are saved in folder "PCR-RFLP_marker_sequences"

10. Extract DNA sequences (here use 8 introns as example)

Run command line:

```
extract_sequences.pl -r ToxoDB-13.0_TgondiiME49.fa -f List -c ME49_introns_coordinates -o
Introns -A ME49
```

Results for intron sequences are saved in folder "Introns"

If the Illumina sequences are to be collected in different batches at different times, and the complete pipeline is used to process data and call SNPs, it is necessary to combine all batches of mapped bam files and run the second part of the pipeline to call SNPs together.

At times, the two parts of the pipeline may be run separately. For example, if the Illumina sequences are collected in different batches at different times, it is only necessary to process the first part to map sequences to reference genome ME49, there is no need to run the second part of the pipeline to call SNPs. However, when all genome sequences from different batches at different times are mapped to the reference genome, it is time to call SNPs from the complete collection of mapped sequences. The second part of the pipeline will pileup the SNPs and generate a SNP summary file "All_SAMPLEs.mpileup.filter.annotation.SNP.summary.txt"

The command line:

```
call_snps_pipeline.pl -r ToxoDB-13.0_TgondiiME49.fa -nosnp F -onlysnp F -f List -delete T -rflp
T -A ME49
```

-nosnp <call SNP> -onlysnp <call SNP only]>

(1) -nonsnp F -onlysnp F, runs the complete pipeline to generate SNP summary file "All_SAMPLEs.mpileup.filter.annotation.SNP.summary.txt"

(2) -nosnp T -onlysnp F, only runs the first part of the pipeline to map sequence reads to reference strain ME49, generates "SRRXXXXXXX_ID.sorted.dedup.realigned.reads.bam" file. No SNPs are called.

(3) -nosnp F, onlysnp T, only runs the second part to pileup SNPs, generate "ALL_SAMPLES.mpileup.filter.annotation.SNP.summary.txt"

I. Run the first part of the pipeline. Generate BAM files (e. g. "SRRXXXXXXX_ID.sorted.dedup.realigned.reads.bam") for different batches of sequences.

In a directory such as "Working_dir1", map Illumina sequences to reference strain ME49.

Run command line:

```
call_snps_pipeline.pl -r ToxoDB-13.0_TgondiiME49.fa -nosnp T -onlysnp F -f List -delete T -rflp T -A ME49
```

Note:

The "List" file contains the SRRXXXXXXX_IDs of strains

This step will not pileup the SNPs for the listed sequences.

Intermediate bam files will be deleted (-delete T) to free up space on hard drive

Fasta files generated for RFLP analysis (-rflp T).

Output:

Generate "SRRXXXXXXX_ID.sorted.dedup.realigned.reads.bam" files

Generate "Bam_List" containing the list of BAM files.

Below is an example in "Bam_List":

```
SRR2068633_EGS.sorted.dedup.realigned.reads.bam
SRR3666803_TgCkBr016.sorted.dedup.realigned.reads.bam
SRR3666806_TgPumaMe1.sorted.dedup.realigned.reads.bam
SRR5643344_TgCkBr01.sorted.dedup.realigned.reads.bam
```

2. Repeat above steps for each batch of sequences ("Working_dir2", "Working_dir3" and so on).

3. When all batches of sequences are mapped to reference ME49, combine all bam files in directory "Woking_all".

Copy and paste all SRRXXXXXXX_ID.sorted.dedup.realigned.reads.bam files from different Working-dir into folder "Working_all".

Copy all sequence IDs into the "List" file and save in folder "Working_all". Example below:

```
SRR2068633_EGS
SRR5643344_TgCkBr01
.................
SRR3666803_TgCkBr016
SRR3666806_TgPumaMe1
```

In "Working_all", make a subfolder "Log" (if not already there), make a file "Snp_call.log" in the "Log" subfolder. Copy and paste records from "Snp_call.log" in each Working_dir into the "Snp_call.log"

All lines listed below should be removed from the new file "Snp_call.log". Examples are below:

```
STEP:Mpileup 23 Nov (Fri) 0:29
STEP:BCF_call 23 Nov (Fri) 0:38
STEP:BCF_filter 23 Nov (Fri) 0:38
STEP:SnpEff 23 Nov (Fri) 0:40
STEP:Summarize_coding_snps 23 Nov (Fri) 0:40
```

The new "Snp_call.log" file should look like this:

```
STEP:Trimmomatic.SRR2068633_EGS 23 Nov (Fri) 10:15
STEP:Trimmomatic.SRR5643344_TgCkBr01 23 Nov (Fri) 10:34
STEP:Bowtie2.SRR2068633_EGS 23 Nov (Fri) 11:34
STEP:Bowtie2.SRR5643344_TgCkBr01 23 Nov (Fri) 14:17
STEP:SortSam.SRR2068633_EGS 23 Nov (Fri) 14:30
STEP:MarkDuplicates.SRR2068633_EGS 23 Nov (Fri) 14:40
STEP:Indexing.SRR2068633_EGS 23 Nov (Fri) 14:40
STEP:Realigning.SRR2068633_EGS 23 Nov (Fri) 15:44
STEP:SortSam.SRR5643344_TgCkBr01 23 Nov (Fri) 16:18
STEP:MarkDuplicates.SRR5643344_TgCkBr01 23 Nov (Fri) 16:38
STEP:Indexing.SRR5643344_TgCkBr01 23 Nov (Fri) 16:39
STEP:Realigning.SRR5643344_TgCkBr01 23 Nov (Fri) 18:35
.................
STEP:Trimmomatic.SRR3666803_TgCkBr016 22 Nov (Thu) 9:47
STEP:Trimmomatic.SRR3666806_TgPumaMe1 22 Nov (Thu) 10:7
STEP:Bowtie2.SRR3666803_TgCkBr016 22 Nov (Thu) 12:46
STEP:Bowtie2.SRR3666806_TgPumaMe1 22 Nov (Thu) 15:24
STEP:SortSam.SRR3666803_TgCkBr016 22 Nov (Thu) 16:1
STEP:MarkDuplicates.SRR3666803_TgCkBr016 22 Nov (Thu) 16:25
STEP:Indexing.SRR3666803_TgCkBr016 22 Nov (Thu) 16:26
STEP:Realigning.SRR3666803_TgCkBr016 22 Nov (Thu) 17:44
STEP:SortSam.SRR3666806_TgPumaMe1 22 Nov (Thu) 18:23
STEP:MarkDuplicates.SRR3666806_TgPumaMe1 22 Nov (Thu) 18:49
STEP:Indexing.SRR3666806_TgPumaMe1 22 Nov (Thu) 18:51
STEP:Realigning.SRR3666806_TgPumaMe1 22 Nov (Thu) 19:57
```

4. Make a subfolder "PCR-RFLP" in folder "Working_all", copy files from each batch of "PCR-RFLP"

II. Run the second part of the pipeline to generate SNP summary file "All_SAMPLEs.mpileup.filter.annotation.SNP.summary.txt"

Run command line:

```
call_snps_pipeline.pl -r ToxoDB-13.0_TgondiiME49.fa -nosnp F -onlysnp T -f List -delete F -rflp
F -A ME49
```

Follow the main protocol to complete data analysis

2.3. RESULTS AND DISCUSSION

Figure 9 shows the phylogenetic network based on SNPs from the 62 whole genomes of *T. gondii* generated from SplitsTree4.

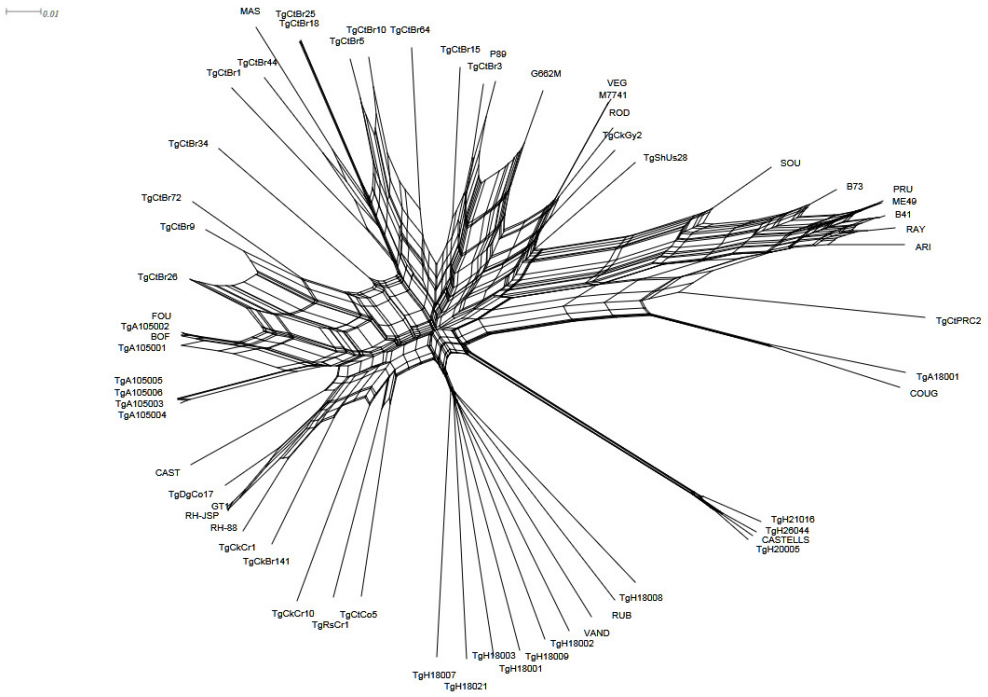


Figure 9: Phylogenetic network analysis of SNPs from the whole genomes of 62 *T. gondii* isolates.

Network based on each chromosome are summarized in Supplementary Figure S1-S15 and the results of FastStructure analysis is presented in Figure 10.

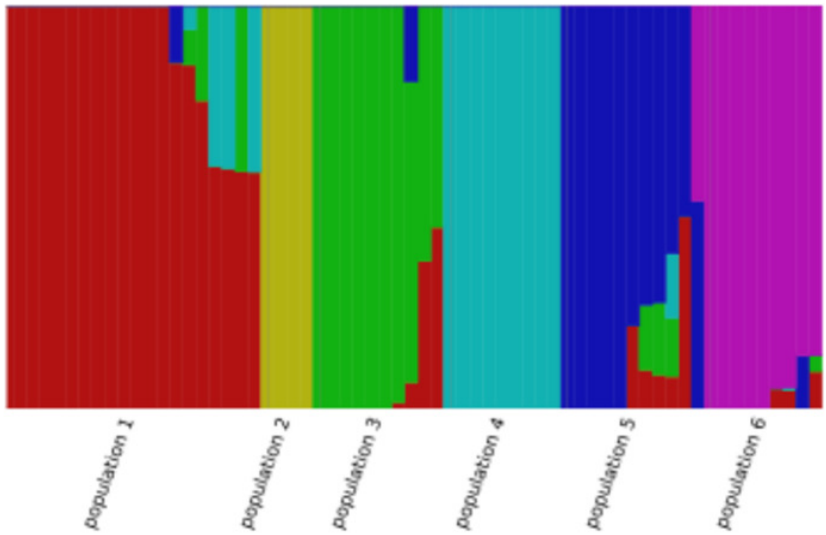


Figure 10: FastStructure analysis of 62 *T. gondii* strains.

The genotype profiles for 62 *T. gondii* isolates were generated for 10 PCR-RFLP genetic markers based on SNPs identified by the pipeline are summarized in Table 1 and all Perl scripts are provided in Supplementary Perl scripts.

In Figure 9, the three archetypal lineages, grouped by Howe and Sibley in 1995 (HOWE; SIBLEY, 1995) are clearly distributed through the phylogenetic tree, indicating that differences in DNA level agree with genotypic and phenotypic differences between these strains. The reference isolate GT1 (Type I) occupies the lower left quadrant, completely opposite the reference ME49 (Type II). The type III isolate (VEG) is positioned at the top of the image. Other strains, such as COUG, MAS, TgCatBr5, TgCatBr64 and TgRsCr1, used as a reference in the PCR-RFLP genotyping method are distributed throughout the network.

The majority of isolates from North America and Europe is grouped into the branches occupied by the archetypal types I, II and III, in agreement with networks generated from the PCR-RFLP genotyping method and confirming a clonal population structure in these continents (HOWE; SIBLEY, 1995). The Brazilian isolates are widely distributed through the phylogenetic tree, occupying basically the entire left side of the

net. This confirms high recombination rates in the parasite population and an epidemic population structure in Brazil with a few expanded clonal lineages (PENA et al., 2008).

The isolates RUB, VAND, TgH18001, TgH18002, TgH18003, TgH18007, TgH18008, TgH18009 and TgH18021 form a well-defined branch, positioned at the bottom of the network and they are all associated with cases of severe systemic toxoplasmosis in immunocompetent adults from French Guiana ().

The analysis from FastStructure software presented in Figure 10, helps to understand how many ancestral genomes are contributing to the genetic diversity of actual *T. gondii* population diversity, where each column represents a different strain. However, it is impossible to identify individuals from the x axis since there is no space between adjacent strains and all strains are presented together. Six different colors are representing different populations and consequently, different ancestral genomes (North America, Central America, South America, Europe, Africa and Asia) for all 62 strains. In this case, it is not possible to identify which strain belong to which population, since the individual strain is not labeled. Some strains are made of different colors, indicating recombination.

As the Virtual Machine was developed, new tools and adaptions were introduced in order to improve the use of the machine and expand the use for users:

- Developed a script to extract any sequences given a coordinates file;
- Developed a script to handle FASTA file, concatenating it and changing the header;
- Developed a script to handle STR (FastStructure file), getting SNP subsets;
- Developed a script to download FASTQ reads, using fastq-dump, given an SRA and rename them, based on List file;
- Improvement on pipeline, generating flags to check all final process, keeping last successful command.

Strain	SAG1	3' SAG2	5' SAG2	alt. SAG2	SAG3	BTUB	GRA6	c22-8	c29-2	L358	PK1	Apico
ARI	u-1	I or II	II	II	II	II	II	II	II	I	II	I
B41	II or III	I or II	II	II	II	II	II	II	II	I	II	I
B73	II or III	I or II	II	II	II	II	II	II	II	III	III	II
BOF	I	I or II	I or III	I	III	I	II	u-1	I	I	I	I
CAST	I	I or II	I or III	I	I	I	I	II	I	III	I	III
CASTELLS	u-1	I or II	I or III	II	III	III	III	III	I	I	III	I
COUG	I	I or II	II	II	III	II	II	II	u-1	I	u-2	I
FOU	I	I or II	I or III	I	III	I	II	u-1	I	I	I	I
G662M	I	III	I or III	III	III	III	III	III	III	III	III	I
GT1	I	I or II	I or III	I	I	I	I	I	I	I	I	I
M7741	II or III	III	I or III	III	III	III	I	III	III	III	III	III
MAS	u-1	I or II	I or III	II	III	III	III	u-1	I	I	III	I
ME49	II or III	I or II	II	II	II	II	II	II	II	II	II	II
p89	I	III	I or III	III	III	III	III	II	III	III	III	III
PRU	II or III	I or II	II	II	II	II	II	II	II	II	II	I
RAY	u-1	I or II	II	II	II	II	II	II	II	I	II	I
RH-88	I	I or II	I or III	I	I	I	I	I	I	I	I	I
RH-JSR	I	I or II	I or III	I	I	I	I	I	I	I	I	I
ROD	I	III	I or III	III	III	III	III	III	III	III	u-2	III
SOU	II or III	I or II	II	II	III	III	III	II	II	III	III	I
TgA105001	I	I or II	I or III	I	III	I	II	u-1	I	I	I	I

Table 1: Genotype profiles for 62 *T. gondii* isolates generated for 10 PCR-RFLP genetic markers.

Strain	SAG1	3' SAG2	5' SAG2	alt. SAG2	SAG3	BTUB	GRA6	c22-8	c29-2	L358	PK1	Apico
TgA105002	I	I or II	I or III	I	III	I	II	u-1	I	I	I	I
TgA105003	I	I or II	I or III	I	III	I	II	II	III	III	I	III
TgA105004	I	I or II	I or III	I	III	I	II	II	III	III	I	III
TgA105005	I	I or II	I or III	I	III	I	II	II	III	III	I	III
TgA105006	I	I or II	I or III	I	III	I	II	II	III	III	I	III
TgA18001	I	I or II	II	II	II	II	II	II	II	I	II	I
TgCtBr1	I	I or II	I or III	II	III	III	III	I	III	I	II	III
TgCtBr10	I	III	I or III	III	III	III	III	I	I	I	III	III
TgCtBr15	I	III	I or III	III	III	III	III	III	I	III	III	III
TgCtBr18	I	III	I or III	III	III	III	II	u-1	I	I	II	I
TgCtBr25	I	III	I or III	III	III	III	II	u-1	I	I	II	I
TgCtBr26	I	I or II	I or III	I	III	I	II	I	I	I	u-1	I
TgCtBr3	I	III	I or III	III	III	III	III	II	III	III	III	III
TgCtBr34	I	I or II	I or III	II	III	I	III	u-1	I	I	III	I
TgCtBr44	u-1	I or II	I or III	II	III	III	III	II	I	I	u-1	I
TgCtBr5	I	III	I or III	III	III	III	III	I	I	I	u-1	I
TgCtBr64	I	I or II	I or III	u-1	III	III	III	u-1	I	III	III	I
TgCatBr72	I	I or II	I or III	I	III	III	II	u-1	I	I	II	I
TgCatBr9	I	I or II	I or III	I	III	III	II	I	I	I	u-1	I
TgCtCo5	I	I or II	I or III	II	III	I	III	II	III	I	u-2	I
TgCatPRC2	u-1	I or II	II	II	III	III	II	II	III	II	II	I

(Continuation)

Strain	SAG1	3' SAG2	5' SAG2	alt. SAG2	SAG3	BTUB	GRA6	c22-8	c29-2	L358	PK1	Apico
TgCkBr141	I	I or II	I or III	I	I	I	I	u-1	I	I	III	III
TgCkCr1	I	I or II	I or III	II	I	I	I	II	I	I	I	I
TgCkCr10	I	I or II	I or III	II	I	II	I	I	I	I	u-1	I
TgCkGy2	I	III	I or III	III	I	I	III	II	III	III	I	III
TgDgCo17	I	III	I or III	III	III	I	I	I	III	I	I	III
TgH18001	I	I or II	I or III	II	I	III	III	II	III	I	III	I
TgH18002	I	I or II	I or III	II	I	III	III	III	I	III	III	I
TgH18003	I	I or II	I or III	II	I	III	III	II	I	I	III	I
TgH18007	I	I or II	I or III	II	I	III	III	I	III	I	III	III
TgH18008	I	I or II	I or III	II	I	III	III	I	I	New allele	u-1	III
TgH18009	I	I or II	I or III	II	I	III	III	III	I	III	III	I
TgH18021	I	I or II	I or III	II	I	III	III	II	I	II	III	III
TgH20005	u-1	I or II	I or III	II	III	III	III	u-1	I	I	III	I
TgH21016	u-1	I or II	I or III	II	III	III	II	III	I	I	III	I
TgH26044	u-1	I or II	I or III	II	III	III	III	u-1	I	I	III	I
TgRsCr1	u-1	I or II	I or III	II	III	I	III	u-2	I	I	III	I
TgShUs28	II or III	III	I or III	III	I	I	I	III	III	III	I	I
VAND	I	I or II	I or III	II	I	III	III	III	I	III	III	I
VEG	II or III	III	I or III	III	III	III	III	III	III	III	III	III

(Continuation)

In this study, we developed a pipeline capable of identifying SNPs from a whole genome, which is composed of a series of programs. In this pipeline, reads obtained from an NCBI database are treated, mapped against a reference genome, and subjected to some processes that involve sorting, duplicating, indexing, and realigning the reads in insertion or deletion (InDels) regions, so that the end of this step, if a BAM extension file is obtained, which allows the identification of the SNPs.

A total of 62 *T. gondii* isolates were processed, representing a very diverse collection coming from a large part of the world. The phylogenetic trees generated from the generated SNPs presented coherence with the phylogenetic distribution that is known today, based on 10 PCR-RFLP markers, which have been widely used for such purpose and identified nearly 200 genotypes worldwide.

To make these tools available to research community, we integrated the pipeline, relevant bioinformatics software and the Perl scripts into a Virtual Machine, which can be downloaded and used by users of any operating system. The idea is that the user does not need to obtain a computer with Linux operating system and does not need to configure it, by installing all software, since Linux Bioinformatics VM already virtualizes the Linux operating system and contains all programs already installed.

The great advantage of using Linux Bioinformatics VM is the portability and the ability to use a machine capable of performing tasks commonly employed in bioinformatics routines through any computer, by simply installing the VirtualBox program and downloading to Linux Bioinformatics VM.

To date, the vast majority of genotyping and population genetics studies of *T. gondii*, begun with the pioneering study by Howe and Sibley (HOWE; SIBLEY, 1995), have used methodologies such as PCR-RFLP (reviewed by SHWAB et al., 2014) and microsatellites. However, due to the cheapness of sequencing technology, more recent and future studies tend to use this methodology.

With the pipeline developed in this study, it will be possible to identify SNPs and manipulate the whole genome, enabling the development of new virulence markers, integration with existing methods such as PCR-RFLP, microsatellite, MLST, among other lines of research.

The pipeline present into machine was developed using *T. gondii* as a model, but may be extended to other pathogens. For this virtual machine to be used not only by experts in the field of Bioinformatics, but also by the scientific community as a whole, we have developed a detailed manual with information ranging from the installation and configuration of the virtual machine, to detailed information of each step. The manual can be downloaded from: .

2.4. SUPPLEMENTARY MATERIAL

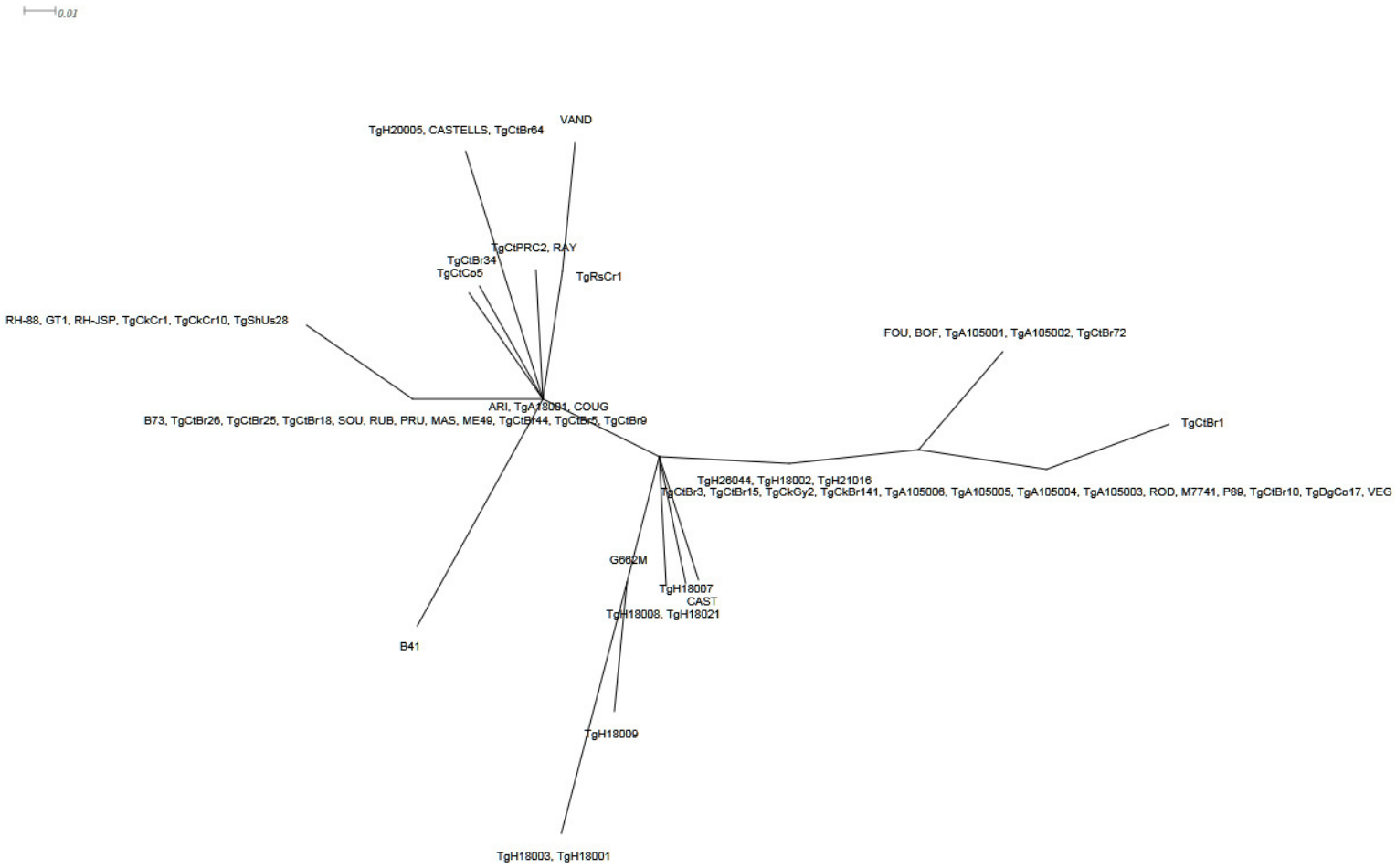


Figure S1: Phylogenetic network analysis of SNPs from Apicoplast of 62 *T. gondii* isolates.

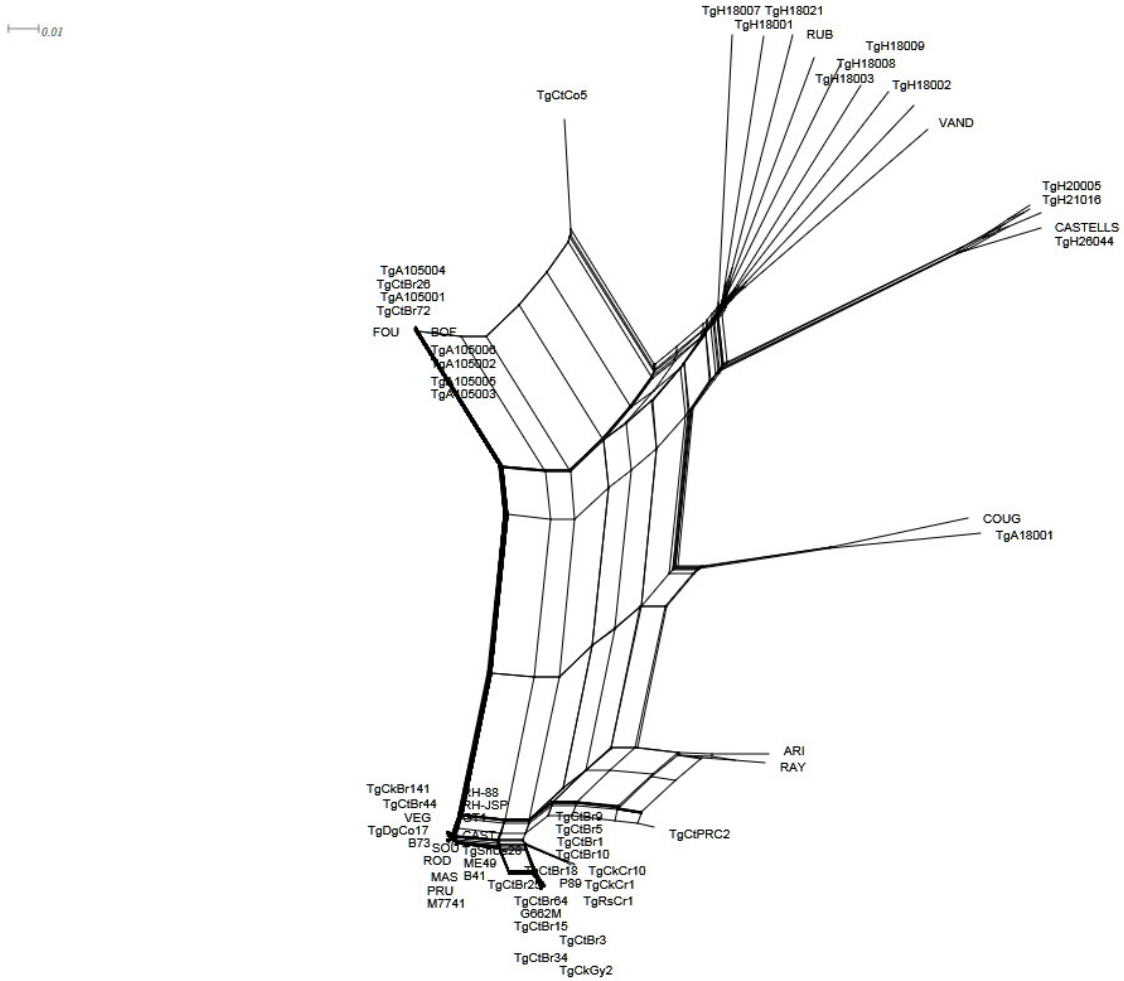


Figure S2: Phylogenetic network analysis of SNPs from chromosome Ia of 62 *T. gondii* isolates.

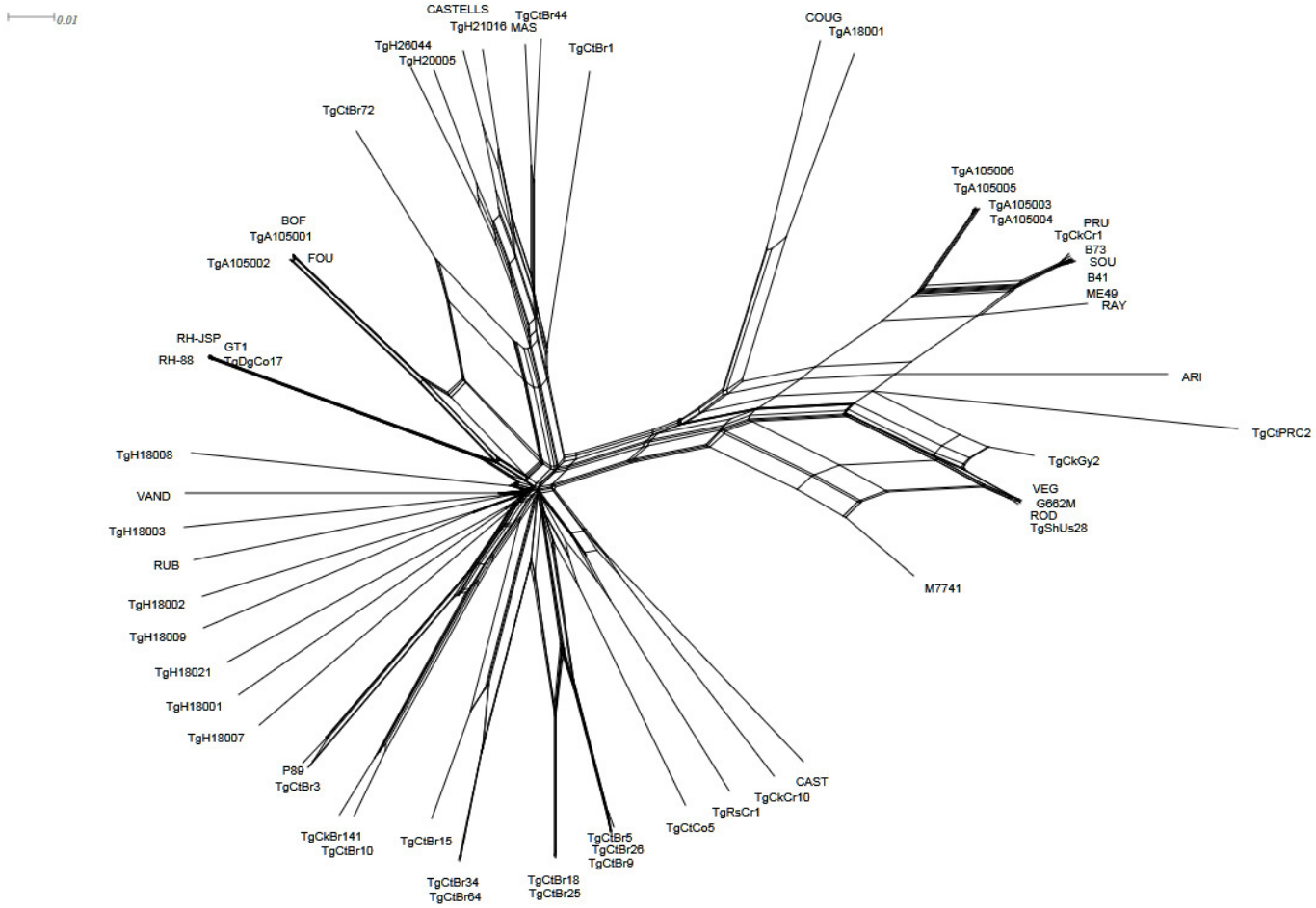


Figure S3: Phylogenetic network analysis of SNPs from chromosome Ib of 62 *T. gondii* isolates.

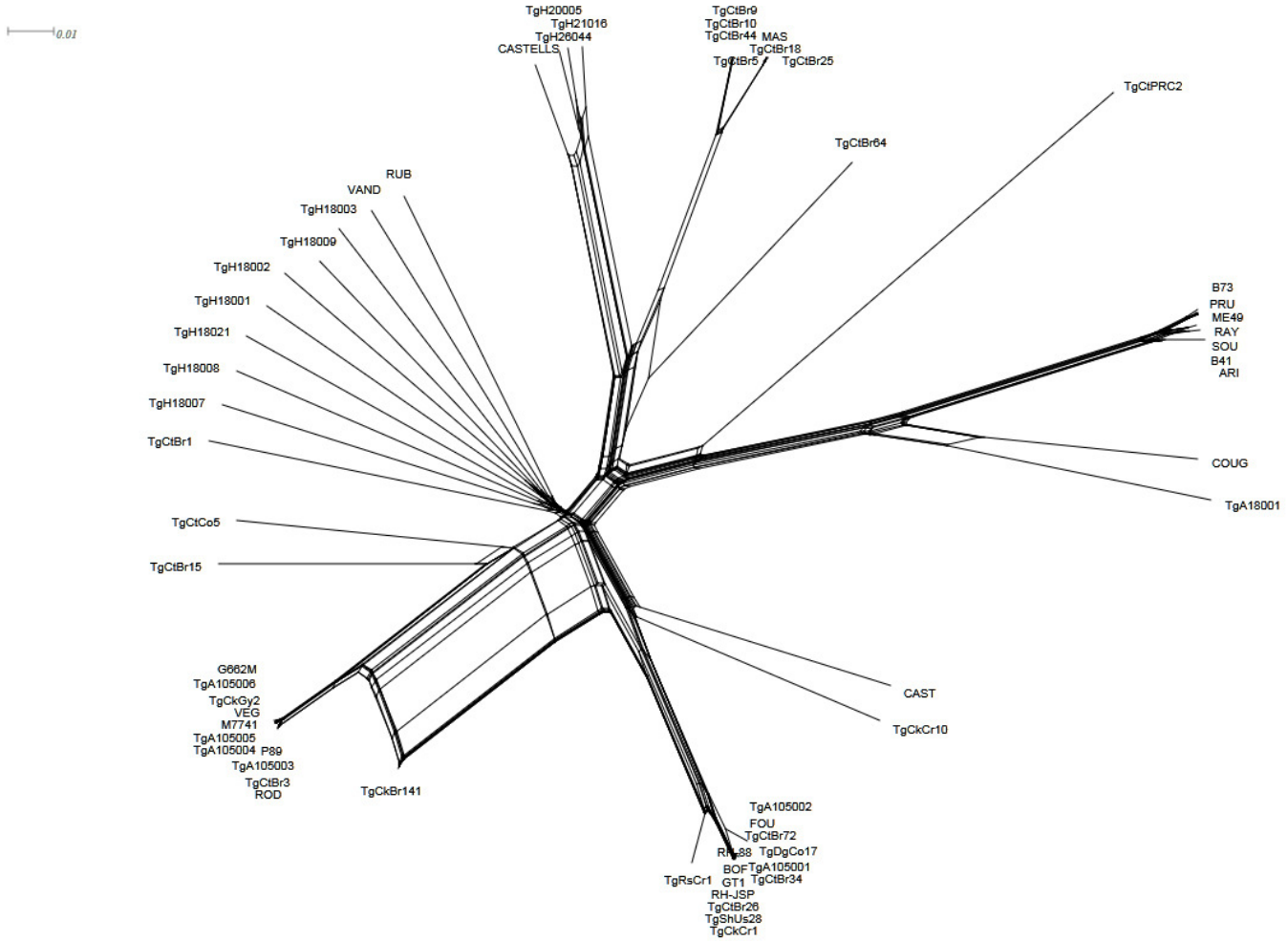


Figure S4: Phylogenetic network analysis of SNPs from chromosome II of 62 *T. gondii* isolates.

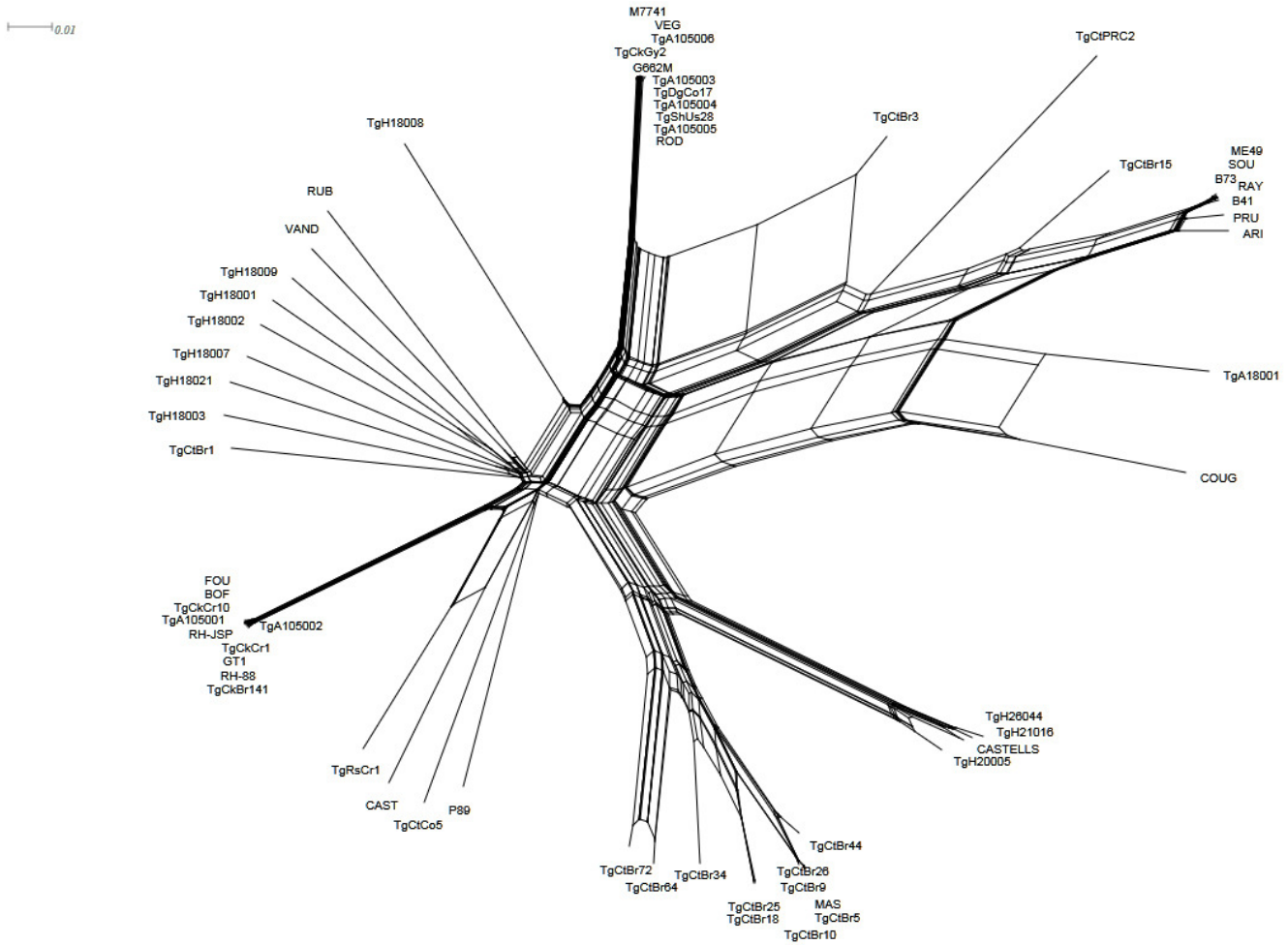


Figure S5: Phylogenetic network analysis of SNPs from chromosome III of 62 *T. gondii* isolates.

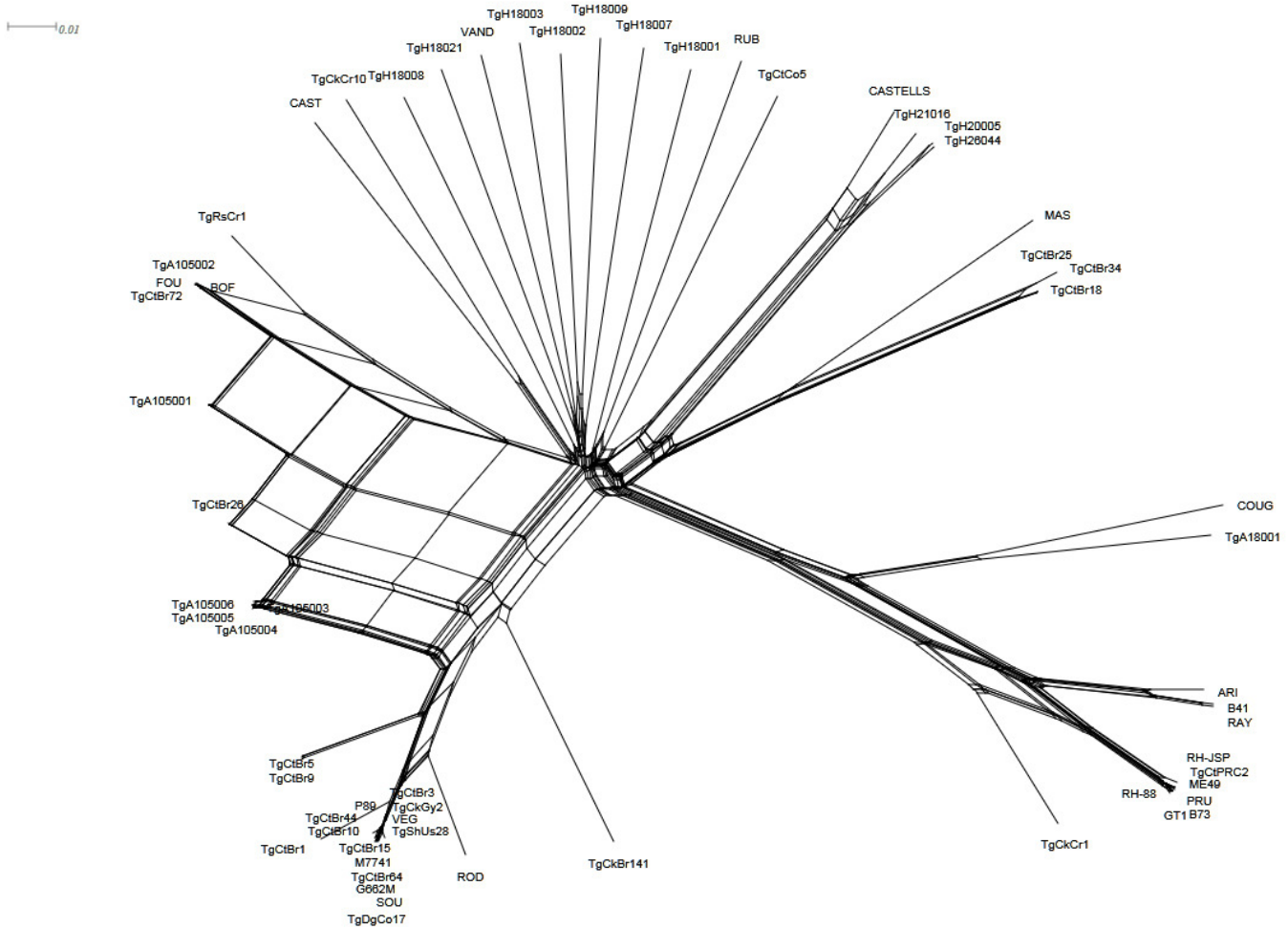


Figure S6: Phylogenetic network analysis of SNPs from chromosome IV of 62 *T. gondii* isolates.

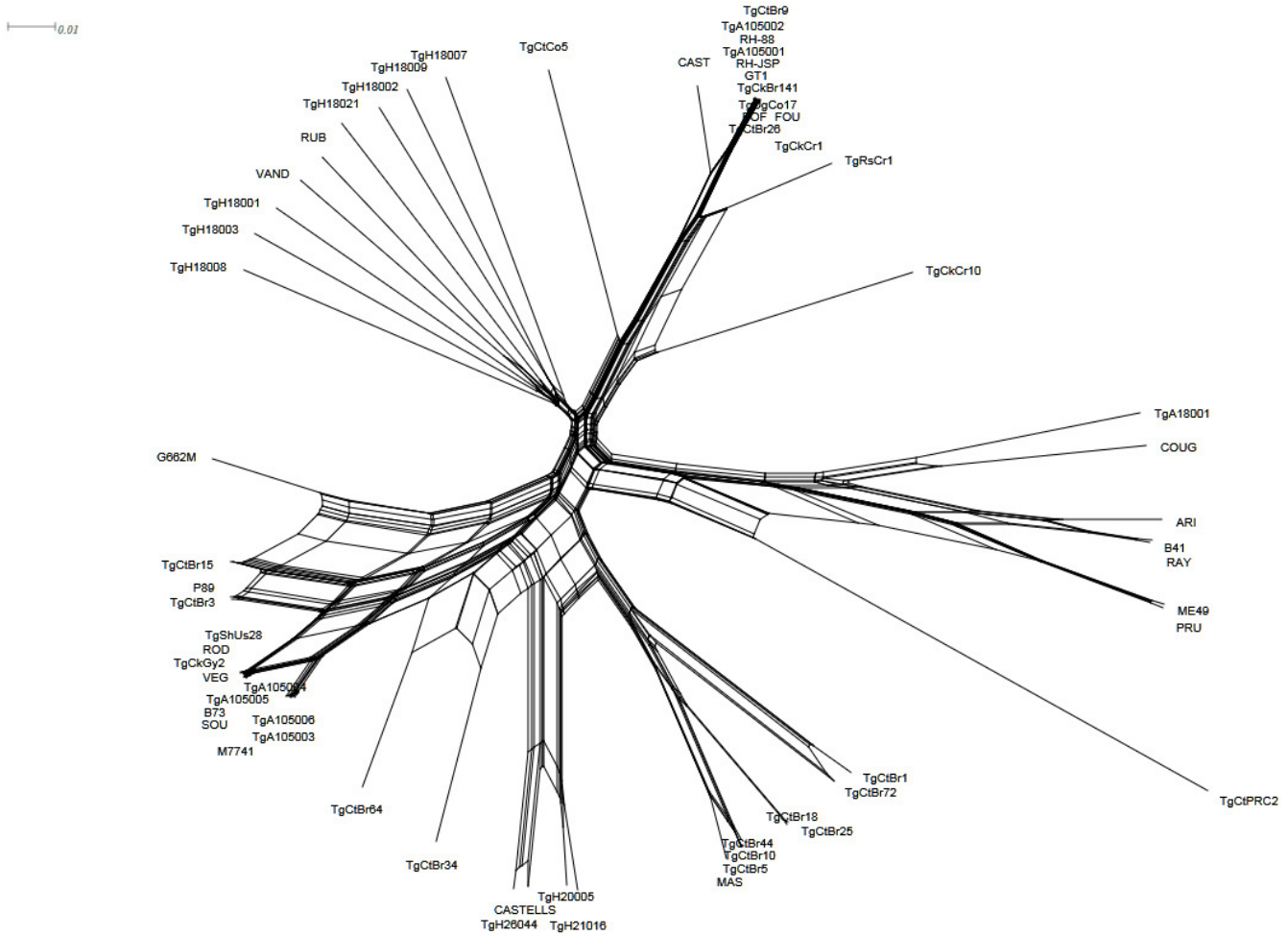


Figure S7: Phylogenetic network analysis of SNPs from chromosome V of 62 *T. gondii* isolates.

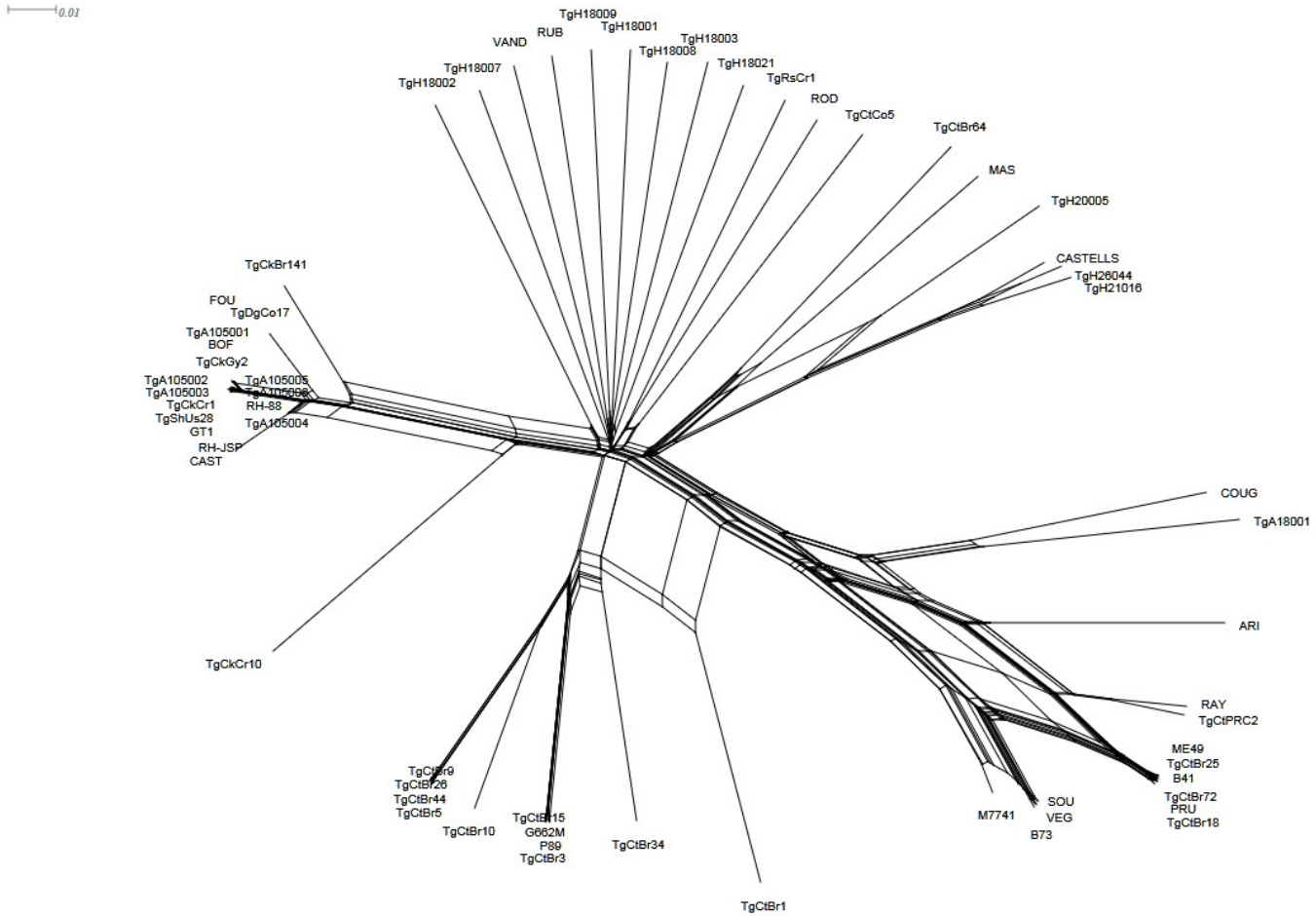


Figure S8: Phylogenetic network analysis of SNPs from chromosome VI of 62 *T. gondii* isolates.

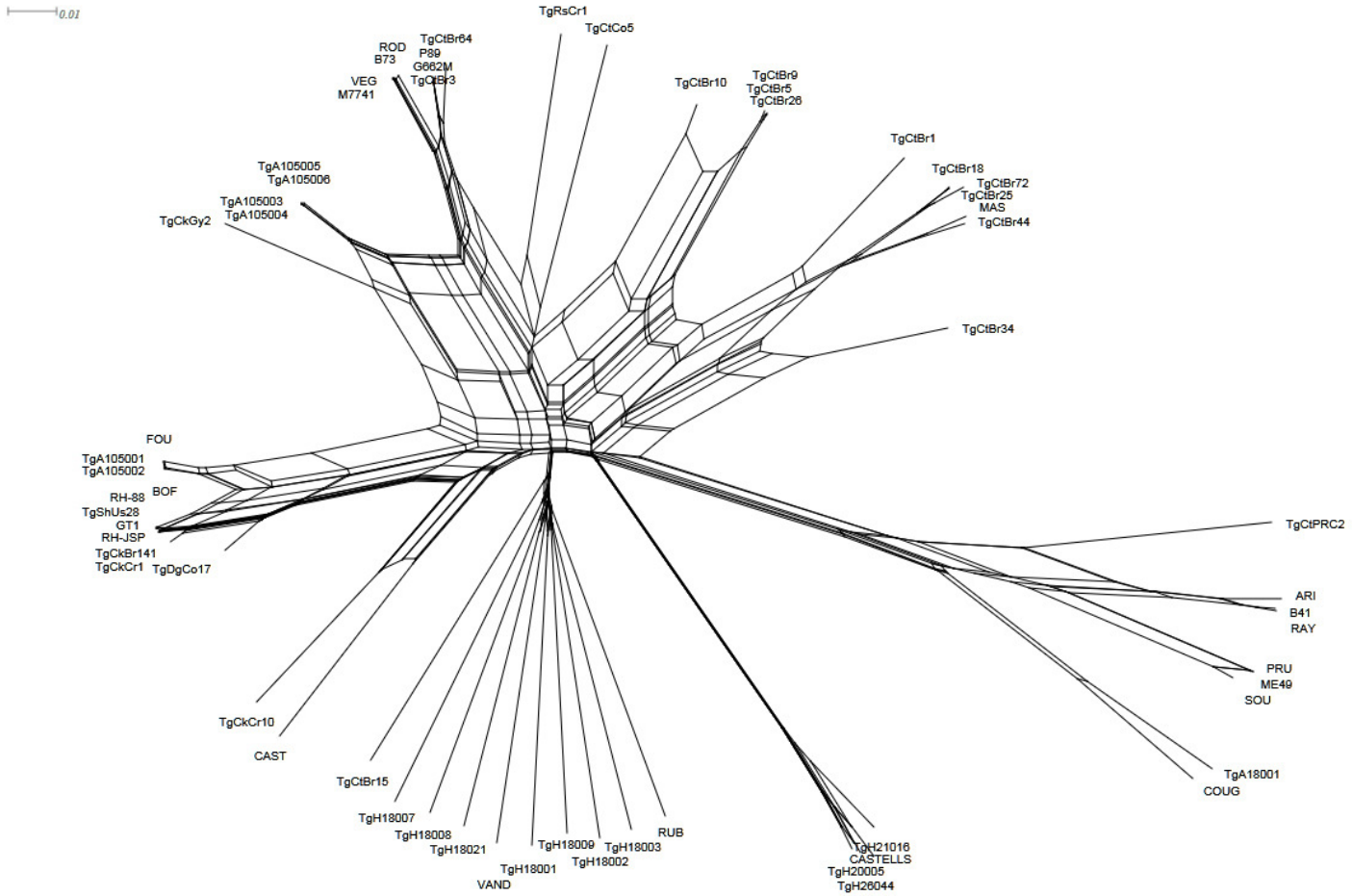


Figure S9: Phylogenetic network analysis of SNPs from chromosome VIIa of 62 *T. gondii* isolates.

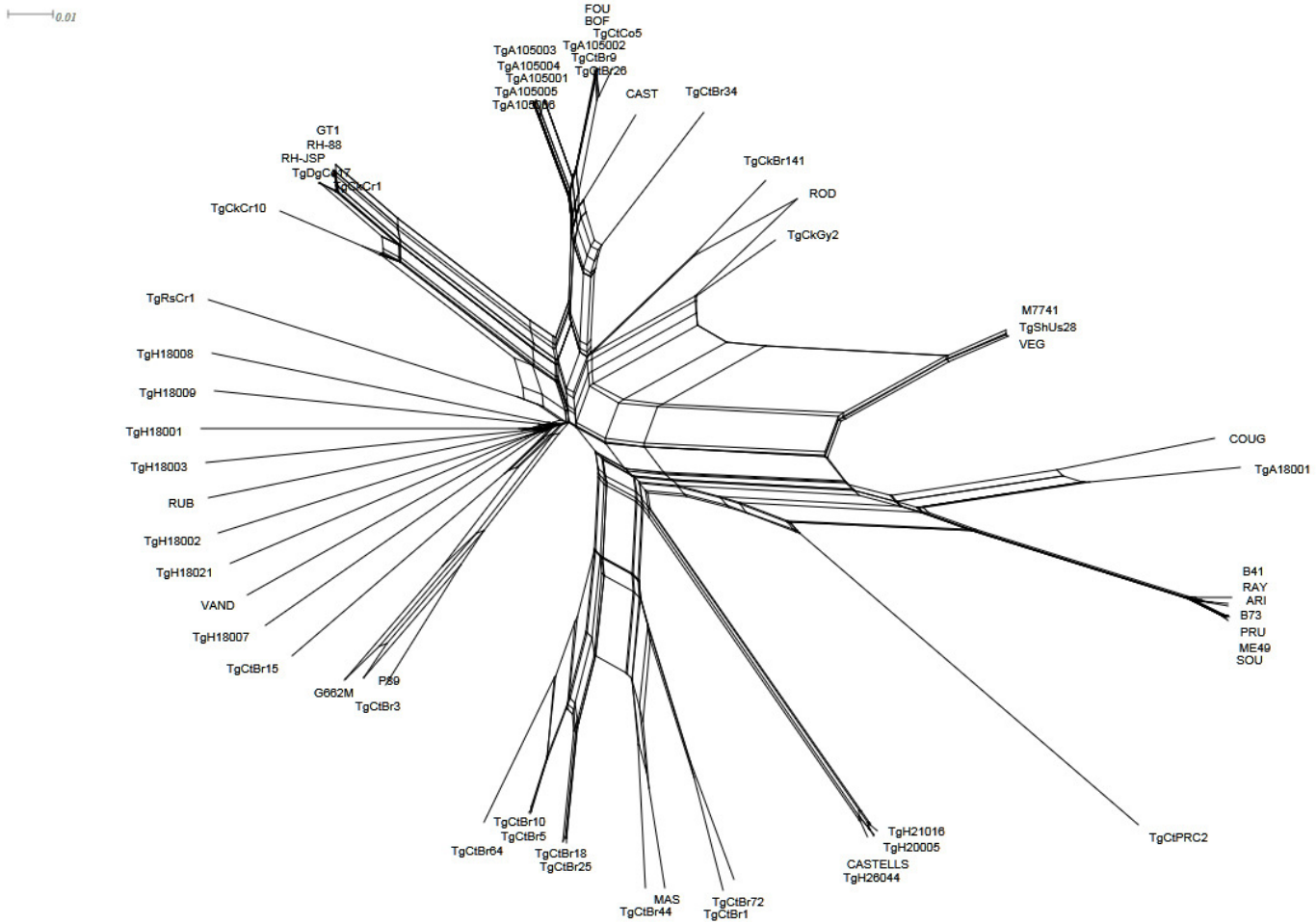


Figure S10: Phylogenetic network analysis of SNPs from chromosome VIIb of 62 *T. gondii* isolates.

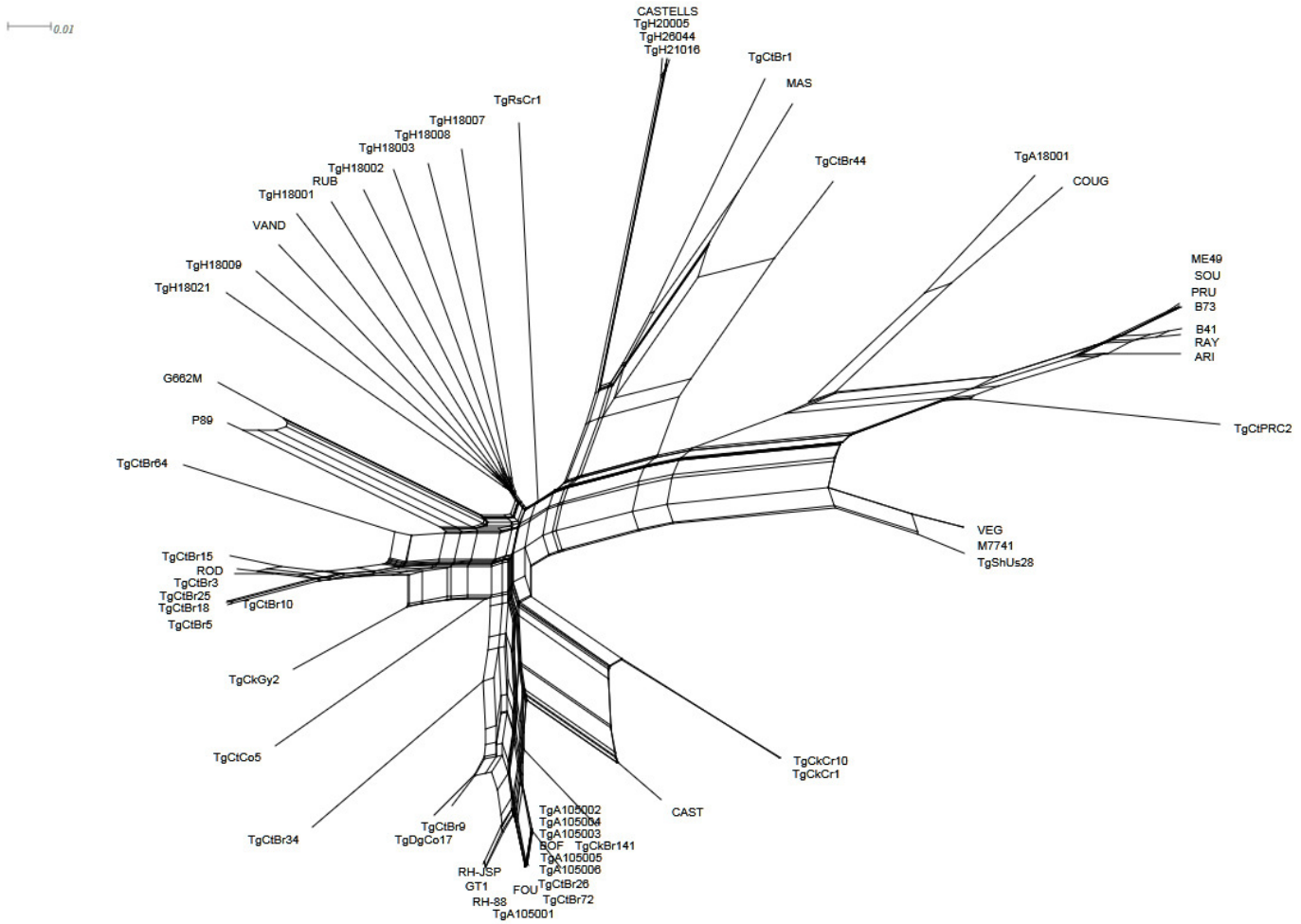


Figure S11: Phylogenetic network analysis of SNPs from chromosome VIII of 62 *T. gondii* isolates.

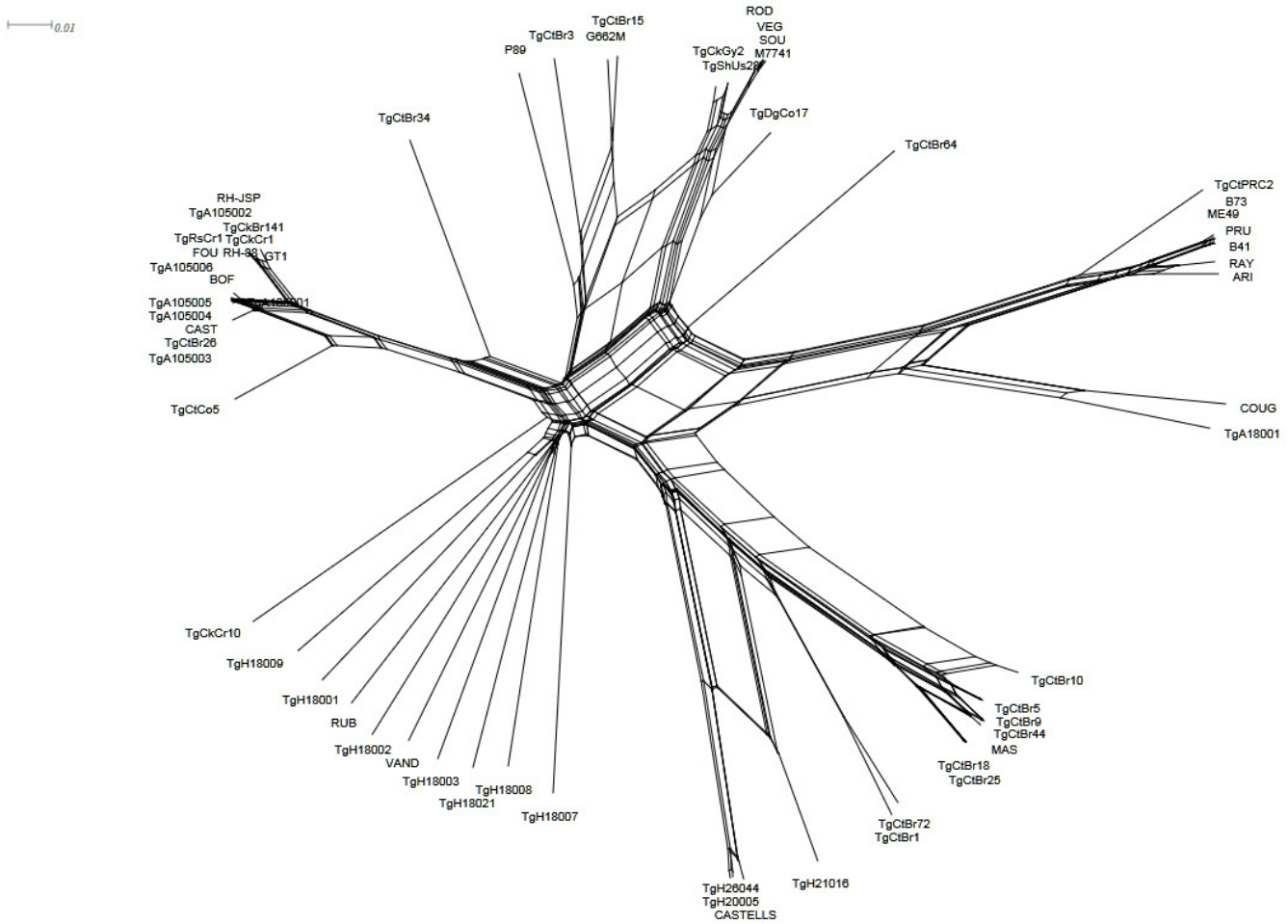


Figure S12: Phylogenetic network analysis of SNPs from chromosome IX of 62 *T. gondii* isolates.

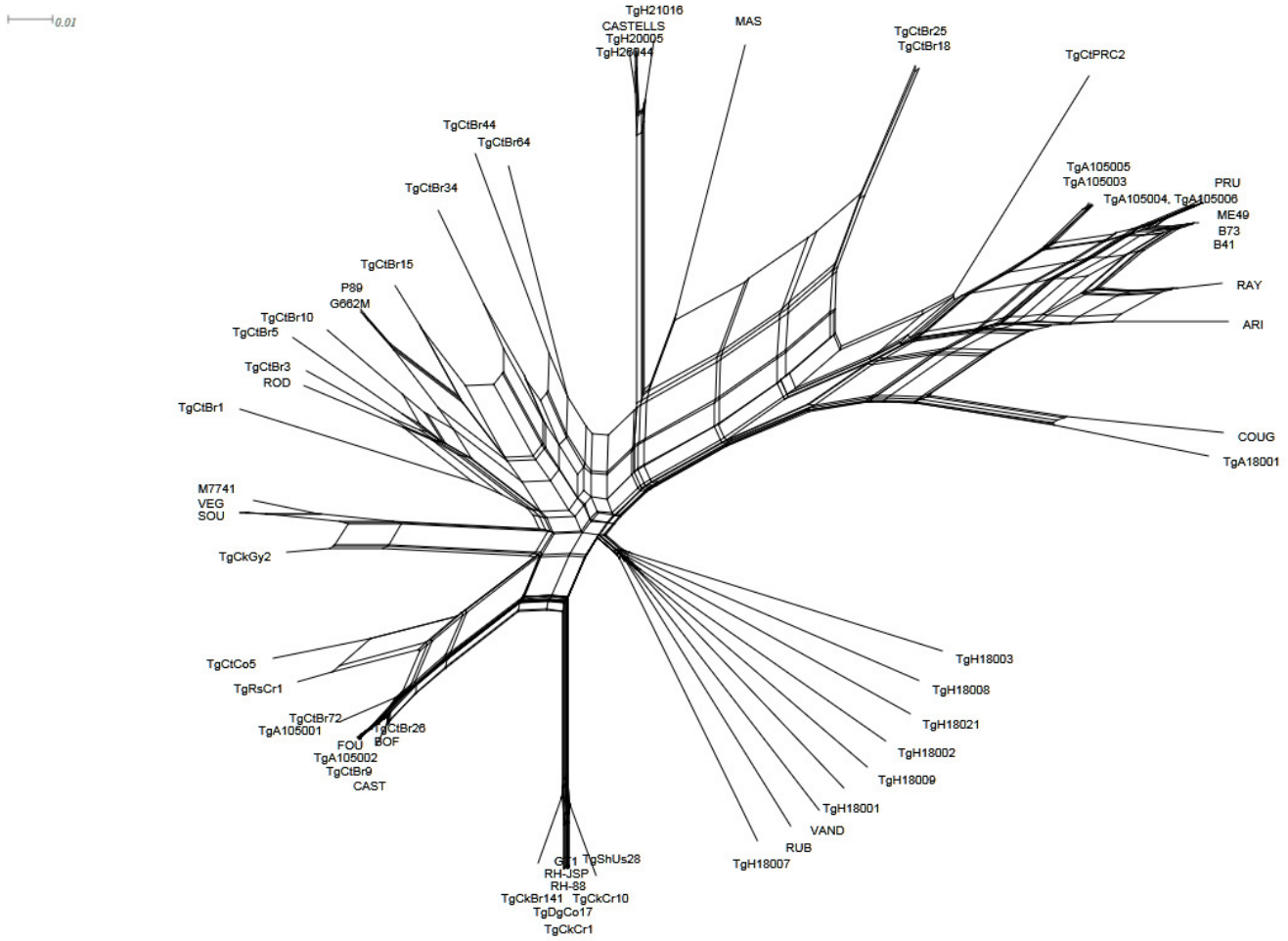


Figure S13: Phylogenetic network analysis of SNPs from chromosome X of 62 *T. gondii* isolates.

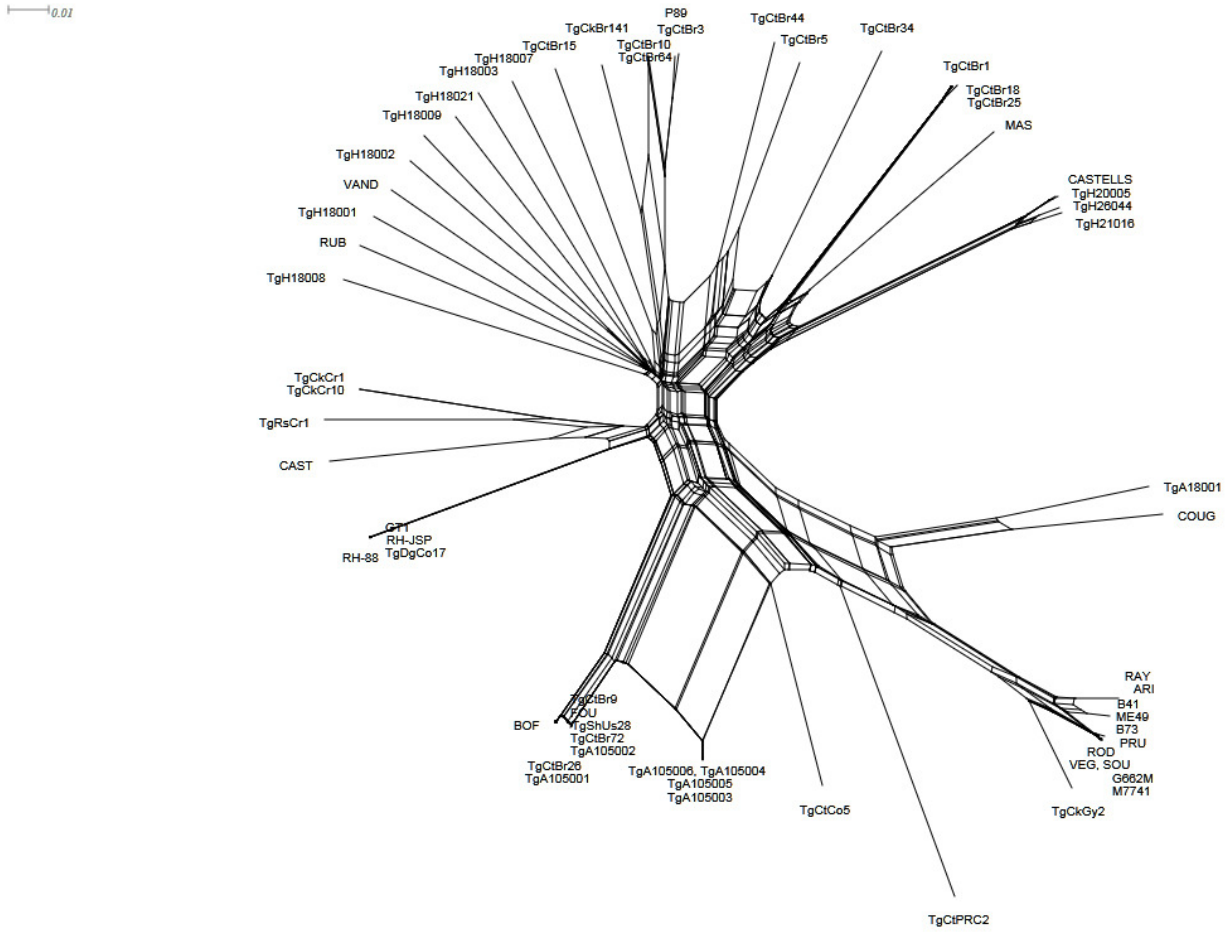


Figure S14: Phylogenetic network analysis of SNPs from chromosome XI of 62 *T. gondii* isolates.

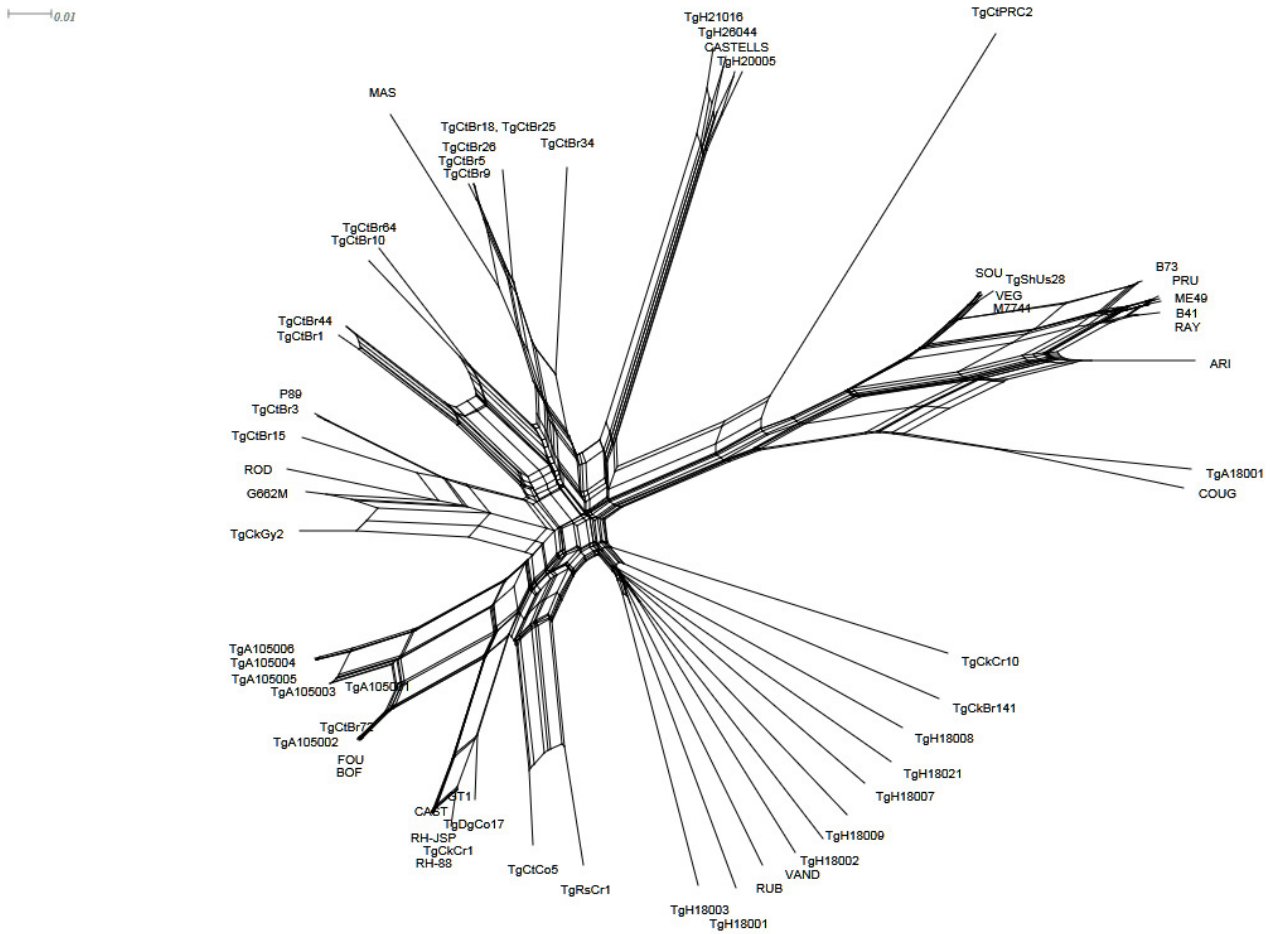


Figure S15: Phylogenetic network analysis of SNPs from chromosome XII of 62 *T. gondii* isolates.

Supplementary Perl script S1: download_sequences.pl source-code

```
#!/usr/bin/perl
use strict;


my $usage = "$0 -f <list with prefixes>\n";
my %arg = @ARGV;
die $usage unless ($arg{-f});

##Defining -f parameter (Load prefixes)
if ($arg{-f})
{
	open (PREFIX, "<$arg{-f}") || die "ERROR, I cannot open $arg{-f}: $!\n\n";
	while(<PREFIX>)
	{
		chomp;
		my %list = split (/\t/, $_);
		foreach my $acc_number (keys %list)
		{
			my $CMD = "fastq-dump --split-3 --gzip -A $acc_number";
			&print_stderr ("##Downloading sequence $acc_number...", $CMD)
unless ((-e $acc_number."_1.fastq.gz") || (-e
$acc_number."_".$list{$acc_number}.".R1.fastq.gz"));
		}
	}
	close PREFIX;

	open (PREFIX, "<$arg{-f}") || die "ERROR, I cannot open $arg{-f}: $!\n\n";
	while(<PREFIX>)
	{
		chomp;
		my %list = split (/\t/, $_);
		foreach my $file (keys %list)
		{
			my $CMD = "mv $file"."_1.fastq.gz
$file"."_"."$list{$file}.R1.fastq.gz";
			&print_stderr ("##Renaming file for $file", $CMD) unless (-e
$file."_".$list{$file}.".R1.fastq.gz");
		}

		foreach my $file (keys %list)
		{
			my $CMD = "mv $file"."_2.fastq.gz
$file"."_"."$list{$file}.R2.fastq.gz";
			&print_stderr ("##Renaming file for $file", $CMD) unless (-e
$file."_".$list{$file}.".R2.fastq.gz");
		}
	}
	close PREFIX;

	if (-e "List")
	{
```

```
		`rm -f "List"`;
	}
	print STDERR "\n\n", "="x100, "\nGenerating List file with prefixes\n",
"="x100, "\n\n";

	open (PREFIX, "<$arg{-f}") || die "ERROR, I cannot open $arg{-f}: $!\n\n";
	while(<PREFIX>)
	{
		chomp;
		my %list = split (/\t/, $_);



		foreach my $strain (keys %list)
		{
			open (LIST, ">>List") || die "ERROR, I cannot open List:
$!\n\n";

			print LIST "$strain"."_"."$list{$strain}\n";
			close LIST;
		}
	}
	print "Done!\n";
}

exit(0);

####################################
sub print_stderr
{
	my ($message, $cmd) = @_;
	print STDERR "\n\n", "="x100, "\n$message\n$cmd\n", "="x100, "\n\n";
	my $err = system("$cmd");
	if ($err)
	{
		die "ERROR, the commands below failed with error $err\n$cmd\n\n";
	}
}
```

Supplementary Perl script S2: call_snps_pipeline.pl source-code

```
#!/usr/bin/perl
use strict;

##Paths
my $data = "/home/user/Bioinformatics/Programs/SnpEff/4.2/Data";
my $programs = "/home/user/Bioinformatics/Programs";

my $GATK = "/home/user/Bioinformatics/Programs/GATK/3.6-0/GenomeAnalysisTK.jar";
my $picard = "/home/user/Bioinformatics/Programs/Picard/2.6.0/Picard.jar";
my $pilon = "/home/user/Bioinformatics/Programs/Pilon/1.22/Pilon.jar";
my $snpEff = "/home/user/Bioinformatics/Programs/SnpEff/4.2/snpEff.jar";
my $trimmomatic =
"/home/user/Bioinformatics/Programs/Trimmomatic/0.36/Trimmomatic.jar";

my $adapters =
"/home/user/Bioinformatics/Programs/Trimmomatic/0.36/Adapters/TruSeq3-PE.fa";
my $summarize_coding_snps =
"/home/user/Bioinformatics/Scripts/summarize_coding_snps.pl";
my $extract_sequences = "extract_sequences.pl";


my $usage = "$0 -r <reference genome> -f <file with other fastq file prefixes> -F
<bcftool filter> -A <reference annotation [GT1/ME49/Pf3D7/ATH other path to
file]> -R <parent/reference fastq prefix if any> -o <oligos file> -S <skip
mappings, just do SNP calls [defaulf F]> -delete <delete temp files [default F]>
-nosnp <call SNP [default F]> -onlysnp <call SNP [default F]> -rflp <extract
sequences to RFLP analysis [default F]>\n";
my %arg = @ARGV;
die $usage unless ($arg{-r} && $arg{-f} && $arg{-A});

my $r = $arg{-r};

##Define -r parameter and Pilon parameters
my $pilon_coordinates;
if ($arg{-r} eq "ME49_apicoplast.fa")
{
	$arg{-r} = "$data/ME49_apicoplast/ME49_apicoplast.fa";
	if ($arg{-crflp})
	{
		$arg{-c} = "/home/user/Bioinformatics/Programs/Pilon/1.22/$arg{-
crflp}";
	}
	else
	{
		$arg{-c} =
"/home/user/Bioinformatics/Programs/Pilon/1.22/Apico_coordinates";
	}
}
elsif ($arg{-r} eq "ToxoDB-13.0_TgondiiME49.fa")
{
	$arg{-r} = "$data/ToxoDB-13.0_TgondiiME49/ToxoDB-13.0_TgondiiME49.fa";
```

```
	if ($arg{-crflp})
	{
		$arg{-c} = "/home/user/Bioinformatics/Programs/Pilon/1.22/$arg{-
crflp}";
	}
	else
	{
		$arg{-c} =
"/home/user/Bioinformatics/Programs/Pilon/1.22/ME49_coordinates";
	}
}

my $bowtie_ref_prefix = $arg{-r};
$bowtie_ref_prefix =~ s/\.(fasta|fa|fsa)$//;
die "ERROR, I cannot find file $arg{-r}: $!\n\n" unless (-e $arg{-r});

##Define -f parameter (Load prefixes)
my @prefix = ($arg{-R}) || ();
if ($arg{-f})
{
	open (PREFIX, "<$arg{-f}") || die "ERROR, I cannot open $arg{-f}: $!\n\n";
	while(<PREFIX>)
	{
		chomp;
		next if m/^[\s\t]*$/;
		s/\s+//g;
		push @prefix, $_;
	}
}

##Define -F parameter
my $filter = "-sLowQual -g3 -G10 -e'%QUAL<10 || (%MAX(DV)<=3 && FMT/GT=\"./1\")||
(%MAX(DV)/%MAX(DP)<=0.3 && FMT/GT=\"./1\") || %MAX(FMT/DP)<5 || (RPB>=0 &&
RPB<0.1 && %QUAL<15) || (MQB>=0 && MQB<0.05) || (BQB>=0 && BQB<0.05) || (MQSB>=0
&& MQSB<0.0)'";
$filter = $arg{-F} if $arg{-F};

##Define -A parameter
my $ref = $arg{-A} eq "GT1" ? "gt1_genome_w_RH_api" : $arg{-A} eq "ME49" ?
"ToxoDB-13.0_TgondiiME49" : $arg{-A} eq "Pf3D7" ? "PlasmoDB_30_Pfalciparum3D7" :
$arg{-A} eq "ATH" ? "athalianaTair10" : "OTHER";

my %annotation = ("GT1" => "$data/Annotations/Tggt1_names.txt", "ME49" =>
"$data/Annotations/Tga4_com_name.txt");
my $ref_annotation_file = $annotation{$arg{-A}} ? $annotation{$arg{-A}} : $arg{-
A};
die "ERROR, I cannot find file $arg{-A}: $!\n\n" unless (-e
$ref_annotation_file);

my %genome = ("ME49" => "$data/ToxoDB-13.0_TgondiiME49/ToxoDB-
13.0_TgondiiME49.fa");
my $ref_genome_file = $genome{$arg{-A}} ? $genome{$arg{-A}} : $arg{-A};
die "ERROR, I cannot find file $arg{-A}: $!\n\n" unless (-e $ref_genome_file);
```

```
my $rflp_folder = $arg{-orflp} ? $arg{-orflp} : "PCR-RFLP";

##Define [default F]
my $delete = $arg{-delete} eq 'T' ? 1 : 0;
my $nosnp = $arg{-nosnp} eq 'T' ? 1 : 0;
my $onlysnp = $arg{-onlysnp} eq 'T' ? 1 : 0;
my $rflp = $arg{-rflp} eq 'T' ? 1 : 0;
my $skip_mapping = $arg{-S} eq 'T' ? 1 : 0;

if ($onlysnp)
{
	$nosnp = 0;
}

##Log file
my %log;
my @months = qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);
my @days = qw(Sun Mon Tue Wed Thu Fri Sat Sun);

##Check log for pipeline status (Restart or resume)
if (-e "Log/Snp_call.log")
{
	open (LOG, "<Log/Snp_call.log");
	while (<LOG>)
	{
		if (m/^STEP:(\S+)/)
		{
			$log{$1} = 1;
		}
	}
}

##Check bowtie2 index files are there, otherwise create them
unless (-e $bowtie_ref_prefix.'.1.bt2')
{
	#warn "I cannot find index files for reference genome $arg{-
r}.\n"."Creating index files with base name: $bowtie_ref_prefix\n" unless
$skip_mapping;
	my $CMD = "bowtie2-build $arg{-r} $bowtie_ref_prefix";
	&print_stderr ("##Building bowtie2 index files with prefix
$bowtie_ref_prefix", $CMD) unless $skip_mapping;
}

##Check dictionary files are there, otherwise create them
unless (-e $bowtie_ref_prefix.'.dict')
{
	my $CMD = "java -jar $picard CreateSequenceDictionary REFERENCE=$arg{-r}
O=$bowtie_ref_prefix.dict";
	&print_stderr ("##Building dict file $bowtie_ref_prefix.dict", $CMD);
}

##Check fai files are there, otherwise create them
unless (-e "$arg{-r}.fai")
{
```

```
	my $CMD = "samtools faidx $arg{-r}";
	&print_stderr ("##Building fai file $arg{-r}.fai", $CMD);
	#my $CMD = "ln -s $arg{-r}.fai $bowtie_ref_prefix.fai";
	#&print_stderr ("##Linking $arg{-r}.fai to $bowtie_ref_prefix.fai", $CMD);
}

my $gz;
my @files_to_map;

unless ($onlysnp)
{
	##Trim FASTQ files
	foreach my $prefix (@prefix)
	{
		next if $skip_mapping;
		my $flag = "Trimmomatic.".$prefix;

		$gz = (-e $prefix.".R2.fastq.gz") ? "fastq.gz" : "fastq";
		##next if (-e $prefix.".paired.R2.$gz");

		my $CMD = "java -jar $trimmomatic PE -threads 10 -phred33
$prefix.R1.$gz $prefix.R2.$gz $prefix.paired.R1.$gz $prefix.unpaired.R1.$gz
$prefix.paired.R2.$gz $prefix.unpaired.R2.$gz ILLUMINACLIP:$adapters:2:30:10
LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:50";
		&print_stderr ("##Trimming reads from $prefix.$gz file", $CMD)
unless ((-e $prefix.".paired.R2.$gz") || $log{$flag});
		&print_log($flag);
		if ($delete)
		{
			`rm -f "$prefix.R1.$gz"`;
			`rm -f "$prefix.R2.$gz"`;
		}
	}

	##Map reads to reference
	foreach my $prefix (@prefix)
	{
		next if $skip_mapping;
		my $flag = "Bowtie2.".$prefix;

		$gz = (-e $prefix.".paired.R2.fastq.gz") ? "fastq.gz" : "fastq";
		##next if (-e $prefix.".bam");

		my $CMD = "bowtie2 --rg-id $prefix --rg PL:Illumina --rg SM:$prefix
--rg LB:$prefix -p 10 --phred33 -x $bowtie_ref_prefix -1 $prefix.paired.R1.$gz -2
$prefix.paired.R2.$gz -U $prefix.unpaired.R1.$gz".",$prefix.unpaired.R2.$gz |
samtools view -bSu - > $prefix.bam";
		&print_stderr ("##Mapping reads from $prefix to reference", $CMD)
unless ((-e $prefix.".bam") || $log{$flag});
		&print_log($flag);
		##I am not deleting the trimmed reads here
	}

	##Eliminate duplicates from BAM/SAM
```

```
	foreach my $prefix (@prefix)
	{
		push @files_to_map, "$prefix.sorted.dedup.realigned.reads.bam";

		##Sort BAM/SAM file
		next if $skip_mapping;
		my $flag = "SortSam.".$prefix;

		my $CMD = "java -Xmx4g -jar $picard SortSam INPUT=$prefix.bam
OUTPUT=$prefix.sorted.bam SO=coordinate TMP_DIR=Temp";
		&print_stderr ("##Sorting $prefix.bam file", $CMD) unless ((-e
"$prefix.sorted.bam") || $log{$flag});
		&print_log($flag);
		if ($delete)
		{
			`rm -f "$prefix.bam"`;
		}
		`rm -rf "Temp"`;

		##Mark duplicated reads
		my $flag = "MarkDuplicates.".$prefix;

		my $CMD = "java -jar $picard MarkDuplicates INPUT=$prefix.sorted.bam
OUTPUT=$prefix.sorted.dedup.bam METRICS_FILE=$prefix.sorted.metrics";
		&print_stderr ("##Marking duplicates in $prefix.sorted.bam file",
$CMD) unless ((-e "$prefix.sorted.dedup.bam") || $log{$flag});
		&print_log($flag);
		if ($delete)
		{
			`rm -f "$prefix.sorted.bam"`;
			`rm -f "$prefix.sorted.bai"`;
			`rm -f "$prefix.sorted.metrics"`;
		}

		##Indexing
		my $flag = "Indexing.".$prefix;

		my $CMD = "samtools index $prefix.sorted.dedup.bam";
		&print_stderr ("##Indexing $prefix.sorted.dedup.bam file", $CMD)
unless ((-e "$prefix.target.intervals.list") || $log{$flag});
		&print_log($flag);
		##I am not deleting $prefix.sorted.dedup.bam file here

		##Realign reads around indels
		my $flag = "Realigning.".$prefix;

		my $CMD = "java -jar $GATK -T RealignerTargetCreator -R $arg{-r} -I
$prefix.sorted.dedup.bam -o $prefix.target.intervals.list";
		&print_stderr ("##Creating target intervals for realign in
$prefix.sorted.dedup.bam file", $CMD) unless ((-e
"$prefix.target.intervals.list") || $log{$flag});
```

```
            my $CMD = "java -Xmx4g -jar $GATK -T IndelRealigner -R $arg{-r} -I
$prefix.sorted.dedup.bam -targetIntervals $prefix.target.intervals.list -o
$prefix.sorted.dedup.realigned.reads.bam";
            &print_stderr ("##Realign reads in $prefix.sorted.dedup.bam file",
$CMD) unless ((-e "$prefix.sorted.dedup.realigned.reads.bam") || $log{$flag});
            &print_log($flag);
            if ($delete)
            {
                  `rm -f "$prefix.sorted.dedup.bam"`;
                  `rm -f "$prefix.sorted.dedup.bai"`;
                  `rm -f "$prefix.target.intervals.list"`;
            }
      }
}

##Check if the PCR-RFLP analysis option is enabled
if ($rflp)
{
      my $CMD = "$extract_sequences -r $r -f $arg{-f} -c $arg{-c} -o
$rflp_folder";
      #my $CMD = "java -jar $pilon --genome $ref_genome_file --jumps
$prefix.sorted.dedup.realigned.reads.bam --output $prefix.rflp --outdir
`pwd`/PCR-RFLP/Original --changes --fix bases --targets $pilon_coordinates\n";
      &print_stderr ("##Extracting PCR-RFLP sequences", $CMD);# unless (-e
"$rflp_folder/Original/$prefix.rflp.fasta");
}

##Generate list with BAM files
      my $CMD = "ls *.sorted.dedup.realigned.reads.bam > Bam_list";
      &print_stderr ("##Generating list with BAM files", $CMD);
      print "Done!\n\n";

##STOP pipeline here if no SNP call required
exit(0) if $nosnp;

foreach my $prefix (@prefix)
{
      push @files_to_map, "$prefix.sorted.dedup.realigned.reads.bam";
}

##Check if file name will be too long
my $p = join("_", @prefix);
$p = length($p) > 50 ? "ALL_SAMPLES" : $p;

##SNP calls with mpileup
my $flag = "Mpileup";

my $CMD = "samtools mpileup -o $p.mpileup.bcf -t DP,DPR,DV,DP4,INFO/DPR,SP -g -s
-u -f $arg{-r} ".join(" ", @files_to_map);
&print_stderr ("##Running mpileup on @files_to_map", $CMD) unless ((-e
"$p.mpileup.bcf") || $log{$flag});
&print_log($flag);

##BCF call
```

```
my $flag = "BCF_call";

my $CMD = "bcftools call -cv -p 0.05 $p.mpileup.bcf > $p.mpileup.vcf";
&print_stderr ("##Running bcftools call on $p.mpileup.bcf", $CMD) unless ((-e
"$p.mpileup.vcf") || $log{$flag});
&print_log($flag);

##BCF filter
my $flag = "BCF_filter";

my $CMD = "bcftools filter $filter $p.mpileup.vcf > $p.mpileup.filter.vcf";
&print_stderr ("##Running bcftools filter on $p.mpileup.vcf", $CMD) unless ((-e
"$p.mpileup.filter.vcf") || $log{$flag});
&print_log($flag);

##SnpEff
my $flag = "SnpEff";

if ($ref eq "OTHER")
{
       print STDERR "No available library for $arg{-A}. SnpEff will not run!\n\n";
       exit(0);
}

my $CMD = "java -Xmx4g -jar $snpEff -ud 1000 $ref $p.mpileup.filter.vcf >
$p.mpileup.filter.annotation.vcf";
&print_stderr ("##Running snpEff on $p.mpileup.filter.vcf", $CMD) unless ((-e
"$p.mpileup.filter.annotation.vcf") || $log{$flag});
&print_log($flag);

my $flag = "Summarize_coding_snps";
my $CMD = "$summarize_coding_snps -i $p.mpileup.filter.annotation.vcf -p F -m F -
n $ref_annotation_file > $p.mpileup.filter.annotation.SNP.summary.txt";
&print_stderr ("##Running summarize_coding_snps.pl on
$p.mpileup.filter.annotation.vcf", $CMD) unless ((-e
"$p.mpileup.filter.annotation.SNP.summary.txt") || $log{$flag});
&print_log($flag);

exit(0);

####################################
sub print_stderr
{
       my ($message, $cmd) = @_;
       print STDERR "\n\n", "="x100, "\n$message\n$cmd\n", "="x100, "\n\n";
       my $err = system("$cmd");
       if ($err)
       {
              die "ERROR, the commands below failed with error $err\n$cmd\n\n";
       }
}

sub print_log
{
```

```
	my $existingdir = './Log';
	mkdir $existingdir unless -d $existingdir;

	my $flag = shift;
	my ($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst) =
localtime();
	open (LOG, ">>Log/Snp_call.log") || die "ERROR, I cannot open Snp_call.log
file: $!\n\n";
	print LOG "STEP:".$flag." $mday $months[$mon] ($days[$wday]) $hour:$min\n";
	close LOG;
}
```

Supplementary Perl script S3: extract_sequences.pl source-code

```
#!/usr/bin/perl
use strict;

##Paths
my $data = "/home/user/Bioinformatics/Programs/SnpEff/4.2/Data";
my $pilon = "/home/user/Bioinformatics/Programs/Pilon/1.22/Pilon.jar";

my $usage = "$0 -r <reference genome> -f <list with prefixes> -c <file with
coordinates> -o <output folder name [default 'Output']>\n";
my %arg = @ARGV;
die $usage unless ($arg{-r} && $arg{-f} && $arg{-c});

##Defining -r parameter
my $ref_prefix = $arg{-r};
$ref_prefix =~ s/\.(fasta|fa|fsa)$//;
$arg{-r} = "$data/$ref_prefix/$arg{-r}";

##Defining -f parameter
my @prefix;
open (IFILE, "<$arg{-f}") || die "ERROR, cannot open file $arg{-f}: $!.\n";
while (<IFILE>)
{
	chomp;
	push @prefix, $_;
}
close IFILE;

##Defining -c parameter
my @row;
my @markers;
open (IFILE, "<$arg{-c}") || die "ERROR, cannot open file $arg{-c}: $!.\n";
open (OFILE, ">$arg{-c}.temp") || die "ERROR, cannot create file $arg{-c}.temp:
$!.\n";
while (<IFILE>)
{
	chomp;
	@row = split ("\t", $_);
	if ($row[1])
	{
		push @markers, $row[1];
	}
	print OFILE "$row[0]\n";
}
close IFILE;
close OFILE;

##Defining -o parameter
$arg = "Output" unless ($arg);

##Script
foreach my $prefix (@prefix)
```

```
{
	##
	unless (-e "$prefix.sorted.dedup.realigned.reads.bai")
	{
		my $CMD = "samtools index $prefix.sorted.dedup.realigned.reads.bam";
		&print_stderr ("##Indexing $prefix.sorted.dedup.realigned.reads.bam
file", $CMD);
		rename ("$prefix.sorted.dedup.realigned.reads.bam.bai",
"$prefix.sorted.dedup.realigned.reads.bai");
	}

	##Extracting sequences from *.bam file
	my $CMD = "java -jar $pilon --genome $arg{-r} --jumps
$prefix.sorted.dedup.realigned.reads.bam --output $prefix --outdir
`pwd`/$arg/Original --changes --fix bases --targets $arg{-c}.temp";
	&print_stderr ("##Extracting sequences for $prefix", $CMD);

	##Generating *.fasta file
	open (IFILE, "<$arg/Original/$prefix.fasta") || die "ERROR, cannot open file
$prefix.fasta: $!.\n";
	open (OFILE, ">$arg/$prefix.fasta") || die "ERROR, cannot create file
$prefix.fasta: $!.\n";
	my $i = 0;
	while (<IFILE>)
	{
		my $row = $_;
		my @id = split ("_", $_);
		if ($row =~ m/^>/)
		{
			if (@markers)
			{
				$id[0] = $prefix;
				$id[1] = $markers[$i];
				print OFILE ">".join ("_", @id);
				$i++;
			}
			else
			{
				$id[0] = $prefix;
				print OFILE ">".join ("_", @id);
			}
		}
		else
		{
			print OFILE "$row";
		}
	}
	close IFILE;
	close OFILE;

	##Generating *.concatenated.fasta file
	open (IFILE, "<$arg/Original/$prefix.fasta") || die "ERROR, cannot open file
$prefix.fasta: $!.\n";
```

```
	open (OFILE, ">$arg/$prefix.concatenated.fasta") || die "ERROR, cannot create
file $prefix.concatenated.fasta: $!.\n";
	my $seq;
	while (<IFILE>)
	{
		my $row = $_;
		unless ($row =~ m/^>/)
		{
			$seq .= $_;
			chomp ($seq);
		}
	}
	print OFILE ">$prefix"."_pilon"."\n$seq\n";
	close IFILE;
	close OFILE;
}
unlink "$arg{-c}.temp";

#################################################
sub print_stderr
{
	my ($msg, $cmd) = @_;
	print STDERR "\n", "="x100, "\n$msg\n$cmd\n", "="x100, "\n";
	my $error = system ("$cmd");
#	if ($error)
#	{
#		die "ERROR, the command below fails with the following error:
$error\n"; #$cmd\n";
#	}
}
```

Supplementary Perl script S4: faststructure.pl source-code

```
#!/usr/bin/perl
use strict;

my $usage = "$0 -i <prefix_input_file>\n";
my %arg = @ARGV;
die $usage unless $arg{-i};

$arg{-i} =~ s/\.[^.]+$//;

##Open an input file
open (INFILE, "<$arg{-i}.fasta") || die "ERROR, I cannot open $arg{-i}: $!\n";

##Create an output file
open (OUTFILE, ">$arg{-i}.str") || die "Can't create file $arg{-i}: $!\n";
open (SAMPLEID, ">$arg{-i}.sample_id") || die "Can't create file $arg{-i}: $!\n";
open (SUBSET01, ">$arg{-i}.SNP_data_01.str") || die "Can't create file $arg{-i}:
$!\n";
open (SUBSET20, ">$arg{-i}.SNP_data_20.str") || die "Can't create file $arg{-i}:
$!\n";
open (SUBSET40, ">$arg{-i}.SNP_data_40.str") || die "Can't create file $arg{-i}:
$!\n";
open (SUBSET60, ">$arg{-i}.SNP_data_60.str") || die "Can't create file $arg{-i}:
$!\n";
open (SUBSET80, ">$arg{-i}.SNP_data_80.str") || die "Can't create file $arg{-i}:
$!\n";


my $id;
my $sequence;

#Create an array to hold the values in each line
while (<INFILE>)
{
	if (m/^>(\S+)/)
	{
		$id = $1;
	}
	else
	{
		$sequence = $_;
		$sequence =~ tr/ATCG/1234/;
		$sequence =~ s/(.)/$1 /g;
		chomp ($sequence);
		chop ($sequence);

		print OUTFILE "$id\t1\t-9\t-9\t-9\t-9\t$sequence\n";
		print OUTFILE "$id\t1\t-9\t-9\t-9\t-9\t$sequence\n";

		print SAMPLEID "$id\t1\t-9\t-9\t-9\t-9\n";

		my $subseq;
```

```
for (my $i = 00; $i <= length($sequence); $i = $i + 100)
{
       my $hundred = substr($sequence, $i, 1);
       $subseq .= $hundred." ";
}
chomp ($subseq);
chop ($subseq);

print SUBSET01 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";
print SUBSET01 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";

my $subseq;
for (my $i = 20; $i <= length($sequence); $i = $i + 100)
{
       my $hundred = substr($sequence, $i, 1);
       $subseq .= $hundred." ";
}
chomp ($subseq);
chop ($subseq);

print SUBSET20 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";
print SUBSET20 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";

my $subseq;
for (my $i = 40; $i <= length($sequence); $i = $i + 100)
{
       my $hundred = substr($sequence, $i, 1);
       $subseq .= $hundred." ";
}
chomp ($subseq);
chop ($subseq);

print SUBSET40 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";
print SUBSET40 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";

my $subseq;
for (my $i = 60; $i <= length($sequence); $i = $i + 100)
{
       my $hundred = substr($sequence, $i, 1);
       $subseq .= $hundred." ";
}
chomp ($subseq);
chop ($subseq);

print SUBSET60 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";
print SUBSET60 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";

my $subseq;
for (my $i = 80; $i <= length($sequence); $i = $i + 100)
{
       my $hundred = substr($sequence, $i, 1);
       $subseq .= $hundred." ";
}
chomp ($subseq);
```

```
            chop ($subseq);

            print SUBSET80 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";
            print SUBSET80 "$id\t1\t-9\t-9\t-9\t-9\t$subseq\n";
      }
}
close INFILE;
close OUTFILE;

close SAMPLEID;
close SUBSET01;
close SUBSET20;
close SUBSET40;
close SUBSET60;
close SUBSET80;
```

Supplementary Perl script S5: rflp_analysis.pl source-code

```
#!/usr/bin/perl
use Bio::Restriction::Analysis;
use Bio::Restriction::Enzyme;
use Bio::Restriction::EnzymeCollection;
use Bio::SeqIO;
use POSIX qw(strftime);
use strict;


my $usage = "$0 -f <list with prefixes>\n";
my %arg = @ARGV;
die $usage unless ($arg{-f});

##Defining -f parameter (Load prefixes)
my @prefix;
if ($arg{-f})
{
	open (PREFIX, "<$arg{-f}") || die "ERROR, I cannot open $arg{-f}: $!\n\n";
	while(<PREFIX>)
	{
		chomp;
		next if m/^[\s\t]*$/;
		s/\s+//g;
		push @prefix, $_;
	}
	close PREFIX;

if (-e "result.rflp.analysis")
{
	`rm -f "result.rflp.analysis"`;
}

	my $datestring = strftime "%e %b %Y - %H:%M:%S", localtime;

	open (OUTFILE, ">>result.rflp.analysis") || die "Can't create file
result.rflp.analysis: $!\n\n";
	print OUTFILE "Strain\tSAG1\t5-SAG2\t3-SAG2\talt-
SAG2\tSAG3\tBTUB\tGRA6\tc22-8\tc29-2\tL358\tPK1\tApico\n";

	open (NEWFILE, ">>new_allele.rflp.analysis") || die "Can't create file
new_allele.rflp.analysis: $!\n\n";
	print NEWFILE "New alleles identified in $datestring\n";

	my $seq;
	foreach my $prefix (@prefix)
	{
		my $seq_in = Bio::SeqIO -> new(-file => "PCR-RFLP/$prefix.fasta");

		my ($sag1, $sag2_5, $sag2_3, $alt_sag2, $sag3, $btub, $gra6, $c22_8,
$c29_2, $l358, $pk1, $apico);
```

```
            my ($sag1_bands, $sag2_5_bands, $sag2_3_bands, $alt_sag2_bands,
$sag3_bands, $btub_bands, $gra6_bands, $c22_8_bands, $c29_2_bands, $l358_bands,
$pk1_bands, $apico_bands);

            while ($seq = $seq_in -> next_seq)
            {
                  #Marker SAG1
                  if ($seq -> length == 390)
                  {
                        my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$seq);

                        my @cuts1 = $analysis -> positions('Sau96I');
                        my @cuts2 = $analysis -> positions('HaeII');

                        my @cuts = (@cuts1, @cuts2);
                        my @sorted_cuts = sort {$b <=> $a} @cuts;

                        my @sizes;
                        for (my $i = 0; $i <= scalar @sorted_cuts; $i++)
                        {
                              if ($i == 0)
                              {
                                    my $fragment = $seq -> length -
$sorted_cuts[$i];
                                    push @sizes, $fragment;
                              }
                              else
                              {
                                    my $fragment = abs($sorted_cuts[$i] -
$sorted_cuts[$i-1]);
                                    push @sizes, $fragment;
                              }
                        }
                        my @sorted_sizes = sort {$b <=> $a} @sizes;
                        $sag1_bands = join(", ", @sorted_sizes);


                        my %type = ("334, 56" => "I", "293, 97" => "II or III",
"237, 97, 56" => "u-1");
                        my $allele = join(", ", @sorted_sizes);
                        if ($type{$allele} ne "")
                        {
                              $sag1 = $type{$allele};
                        }
                        else
                        {
                              $sag1 = "New allele";

                              print NEWFILE "$prefix\n";
                              print NEWFILE "SAG1\t";
                              print NEWFILE join(", ", @sorted_sizes)."\t";
                              print NEWFILE "New allele\n\n";
                        }
```

```
		}

		#Marker 3'-SAG2
		elsif ($seq -> length == 222 || $seq -> length == 221)
		{
			my $rev_com = $seq -> revcom();
			my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$rev_com);

			my @fragments = $analysis -> fragments('HhaI');
			my @sizes = $analysis -> sizes('HhaI');

			my @sorted_sizes = sort {$b <=> $a} @sizes;
			$sag2_3_bands = join(", ", @sorted_sizes);


			my %type = ("222" => "I or III", "221" => "I or III",
"167, 55" => "II");
			my $allele = join(", ", @sorted_sizes);
			if ($type{$allele} ne "")
			{
				$sag2_3 = $type{$allele};
			}
			else
			{
				$sag2_3 = "New allele";

				print NEWFILE "$prefix\n";
				print NEWFILE "3-SAG2\t";
				print NEWFILE join(", ", @sorted_sizes)."\t";
				print NEWFILE "New allele\n\n";
			}
		}

		#Marker 5'-SAG2
		elsif ($seq -> length == 242)
		{
			my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$seq);

			my @fragments = $analysis -> fragments('MboI');
			my @sizes = $analysis -> sizes('MboI');

			my @sorted_sizes = sort {$b <=> $a} @sizes;
			$sag2_5_bands = join(", ", @sorted_sizes);


			my %type = ("242" => "I or II", "186, 56" => "III");
			my $allele = join(", ", @sorted_sizes);
			if ($type{$allele} ne "")
			{
				$sag2_5 = $type{$allele};
			}
			else
```

```
                        {
                                $sag2_5 = "New allele";

                                print NEWFILE "$prefix\n";
                                print NEWFILE "5-SAG2\t";
                                print NEWFILE join(", ", @sorted_sizes)."\t";
                                print NEWFILE "New allele\n\n";
                        }
                }

                #Marker alt-SAG2
                elsif ($seq -> length == 546)
                {
                        my $rev_com = $seq -> revcom();
                        my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$rev_com);

                        my @cuts1 = $analysis -> positions('TaqI');
                        my @cuts2 = $analysis -> positions('HinfI');

                        my @cuts = (@cuts1, @cuts2);
                        my @sorted_cuts = sort {$b <=> $a} @cuts;

                        my @sizes;
                        for (my $i = 0; $i <= scalar @sorted_cuts; $i++)
                        {
                                if ($i == 0)
                                {
                                        my   $fragment   =   $seq   ->   length   -
$sorted_cuts[$i];
                                        push @sizes, $fragment;
                                }
                                else
                                {
                                        my  $fragment  =  abs($sorted_cuts[$i]  -
$sorted_cuts[$i-1]);
                                        push @sizes, $fragment;
                                }
                        }
                        my @sorted_sizes = sort {$b <=> $a} @sizes;
                        $alt_sag2_bands = join(", ", @sorted_sizes);


                        my %type = ("343, 161, 36, 4, 2" => "I", "343, 112, 49,
36, 4, 2" => "II", "379, 161, 4, 2" => "III", "394, 112, 36, 4" => "u-1");
                        my $allele = join(", ", @sorted_sizes);
                        if ($type{$allele} ne "")
                        {
                                $alt_sag2 = $type{$allele};
                        }
                        else
                        {
                                $alt_sag2 = "New allele";
```

```
                        print NEWFILE "$prefix\n";
                        print NEWFILE "alt-SAG2\t";
                        print NEWFILE join(", ", @sorted_sizes)."\t";
                        print NEWFILE "New allele\n\n";
                }
        }

        #Marker SAG3
        elsif ($seq -> length == 226)
        {
                my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$seq);

                my @fragments = $analysis -> fragments('NciI');
                my @sizes = $analysis -> sizes('NciI');

                my @sorted_sizes = sort {$b <=> $a} @sizes;
                $sag3_bands = join(", ", @sorted_sizes);


                my %type = ("99, 65, 62" => "I", "226" => "II", "161,
65" => "III");
                my $allele = join(", ", @sorted_sizes);
                if ($type{$allele} ne "")
                {
                        $sag3 = $type{$allele};
                }
                else
                {
                        $sag3 = "New allele";

                        print NEWFILE "$prefix\n";
                        print NEWFILE "SAG3\t";
                        print NEWFILE join(", ", @sorted_sizes)."\t";
                        print NEWFILE "New allele\n\n";
                }
        }

        #Marker BTUB
        elsif ($seq -> length == 411)
        {
                my $rev_com = $seq -> revcom();
                my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$rev_com);

                my @cuts1 = $analysis -> positions('TaqI');
                my @cuts2 = $analysis -> positions('BsiEI');

                my @cuts = (@cuts1, @cuts2);
                my @sorted_cuts = sort {$b <=> $a} @cuts;

                my @sizes;
                for (my $i = 0; $i <= scalar @sorted_cuts; $i++)
                {
```

```
                                if ($i == 0)
                                {
                                        my   $fragment   =   $seq   ->   length   -
$sorted_cuts[$i];
                                        push @sizes, $fragment;
                                }
                                else
                                {
                                        my   $fragment   =   abs($sorted_cuts[$i]   -
$sorted_cuts[$i-1]);
                                        push @sizes, $fragment;
                                }
                        }
                        my @sorted_sizes = sort {$b <=> $a} @sizes;
                        $btub_bands = join(", ", @sorted_sizes);

                        my %type = ("220, 118, 73" => "I", "191, 127, 93" =>
"II", "127, 118, 93, 73" => "III");
                        my $allele = join(", ", @sorted_sizes);
                        if ($type{$allele} ne "")
                        {
                                $btub = $type{$allele};
                        }
                        else
                        {
                                $btub = "New allele";

                                print NEWFILE "$prefix\n";
                                print NEWFILE "BTUB\t";
                                print NEWFILE join(", ", @sorted_sizes)."\t";
                                print NEWFILE "New allele\n\n";
                        }
                }

                #Marker GRA6
                elsif ($seq -> length == 344)
                {
                        my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$seq);

                        my @fragments = $analysis -> fragments('MseI');
                        my @sizes = $analysis -> sizes('MseI');

                        my @sorted_sizes = sort {$b <=> $a} @sizes;
                        $gra6_bands = join(", ", @sorted_sizes);


                        my %type = ("260, 84" => "I", "181, 163" => "II", "163,
97, 84" => "III");
                        my $allele = join(", ", @sorted_sizes);
                        if ($type{$allele} ne "")
                        {
                                $gra6 = $type{$allele};
                        }
```

```
                        else
                        {
                                $gra6 = "New allele";

                                print NEWFILE "$prefix\n";
                                print NEWFILE "GRA6\t";
                                print NEWFILE join(", ", @sorted_sizes)."\t";
                                print NEWFILE "New allele\n\n";
                        }
                }

                #Marker c22-8
                elsif ($seq -> length == 521)
                {
                        my $rev_com = $seq -> revcom();
                        my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$rev_com);

                        my @cuts1 = $analysis -> positions('BsmAI');
                        my @cuts2 = $analysis -> positions('MboII');

                        my @cuts = (@cuts1, @cuts2);
                        my @sorted_cuts = sort {$b <=> $a} @cuts;

                        my @sizes;
                        for (my $i = 0; $i <= scalar @sorted_cuts; $i++)
                        {
                                if ($i == 0)
                                {
                                        my   $fragment   =   $seq   ->   length   -
$sorted_cuts[$i];
                                        push @sizes, $fragment;
                                }
                                else
                                {
                                        my   $fragment   =   abs($sorted_cuts[$i]   -
$sorted_cuts[$i-1]);
                                        push @sizes, $fragment;
                                }
                        }
                        my @sorted_sizes = sort {$b <=> $a} @sizes;
                        $c22_8_bands = join(", ", @sorted_sizes);


                        my %type = ("175, 156, 112, 59, 19" => "I", "234, 156,
112, 19" => "II", "253, 156, 112" => "III", "175, 156, 112, 78" => "u-1", "390,
112, 19" => "u-2");
                        my $allele = join(", ", @sorted_sizes);
                        if ($type{$allele} ne "")
                        {
                                $c22_8 = $type{$allele};
                        }
                        else
                        {
```

```
                        $c22_8 = "New allele";

                        print NEWFILE "$prefix\n";
                        print NEWFILE "c22-8\t";
                        print NEWFILE join(", ", @sorted_sizes)."\t";
                        print NEWFILE "New allele\n\n";
                }
        }

        #Marker c29-2
        elsif ($seq -> length == 446)
        {
                my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$seq);

                my @cuts1 = $analysis -> positions('RsaI');
                my @cuts2 = $analysis -> positions('HpyCH4IV');

                my @cuts = (@cuts1, @cuts2);
                my @sorted_cuts = sort {$b <=> $a} @cuts;

                my @sizes;
                for (my $i = 0; $i <= scalar @sorted_cuts; $i++)
                {
                        if ($i == 0)
                        {
                                my   $fragment   =   $seq   ->   length   -
$sorted_cuts[$i];

                                push @sizes, $fragment;
                        }
                        else
                        {
                                my  $fragment  =  abs($sorted_cuts[$i]  -
$sorted_cuts[$i-1]);

                                push @sizes, $fragment;
                        }
                }
                my @sorted_sizes = sort {$b <=> $a} @sizes;
                $c29_2_bands = join(", ", @sorted_sizes);


                my %type = ("307, 50, 45, 41, 3" => "I", "165, 142, 53,
45, 41" => "II", "348, 50, 45, 3" => "III", "183, 165, 50, 45, 3" => "u-1");
                my $allele = join(", ", @sorted_sizes);
                if ($type{$allele} ne "")
                {
                        $c29_2 = $type{$allele};
                }
                else
                {
                        $c29_2 = "New allele";

                        print NEWFILE "$prefix\n";
                        print NEWFILE "c29-2\t";
```

```
                        print NEWFILE join(", ", @sorted_sizes)."\t";
                        print NEWFILE "New allele\n\n";
                }
        }

        #Marker L358
        elsif ($seq -> length == 417)
        {
                my $rev_com = $seq -> revcom();
                my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$rev_com);

                my @cuts1 = $analysis -> positions('NlaIII');
                my @cuts2 = $analysis -> positions('HaeIII');

                my @cuts = (@cuts1, @cuts2);
                my @sorted_cuts = sort {$b <=> $a} @cuts;

                my @sizes;
                for (my $i = 0; $i <= scalar @sorted_cuts; $i++)
                {
                        if ($i == 0)
                        {
                                my   $fragment   =   $seq   ->   length   -
$sorted_cuts[$i];
                                push @sizes, $fragment;
                        }
                        else
                        {
                                my  $fragment  =  abs($sorted_cuts[$i]  -
$sorted_cuts[$i-1]);
                                push @sizes, $fragment;
                        }
                }
                my @sorted_sizes = sort {$b <=> $a} @sizes;
                $l358_bands = join(", ", @sorted_sizes);


                my %type = ("281, 48, 34, 27, 23, 4" => "I", "205, 110,
48, 27, 23, 4" => "II", "171, 110, 48, 34, 27, 23, 4" => "III");
                my $allele = join(", ", @sorted_sizes);
                if ($type{$allele} ne "")
                {
                        $l358 = $type{$allele};
                }
                else
                {
                        $l358 = "New allele";

                        print NEWFILE "$prefix\n";
                        print NEWFILE "L358\t";
                        print NEWFILE join(", ", @sorted_sizes)."\t";
                        print NEWFILE "New allele\n\n";
                }
```

```
		}

		#Marker PK1
		elsif ($seq -> length == 902 || $seq -> length == 903)
		{
			my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$seq);

			my @cuts1 = $analysis -> positions('RsaI');
			my @cuts2 = $analysis -> positions('AvaI');

			my @cuts = (@cuts1, @cuts2);
			my @sorted_cuts = sort {$b <=> $a} @cuts;

			my @sizes;
			for (my $i = 0; $i <= scalar @sorted_cuts; $i++)
			{
				if ($i == 0)
				{
					my $fragment = $seq -> length -
$sorted_cuts[$i];
					push @sizes, $fragment;
				}
				else
				{
					my $fragment = abs($sorted_cuts[$i] -
$sorted_cuts[$i-1]);
					push @sizes, $fragment;
				}
			}
			my @sorted_sizes = sort {$b <=> $a} @sizes;
			$pk1_bands = join(", ", @sorted_sizes);


			my %type = ("380, 263, 125, 100, 35" => "I", "237, 215,
143, 125, 100, 48, 35" => "II", "379, 215, 125, 100, 48, 35" => "III", "380, 215,
125, 100, 48, 35" => "III", "353, 125, 108, 107, 100, 48, 35, 27" => "u-1", "353,
215, 125, 100, 48, 35, 27" => "u-2");
			my $allele = join(", ", @sorted_sizes);
			if ($type{$allele} ne "")
			{
				$pk1 = $type{$allele};
			}
			else
			{
				$pk1 = "New allele";

				print NEWFILE "$prefix\n";
				print NEWFILE "PK1\t";
				print NEWFILE join(", ", @sorted_sizes)."\t";
				print NEWFILE "New allele\n\n";
			}
		}
```

```
                        #Marker Apico
                        elsif ($seq -> length == 639)
                        {
                                my $rev_com = $seq -> revcom();
                                my $analysis = Bio::Restriction::Analysis -> new(-seq =>
$rev_com);

                                my @cuts1 = $analysis -> positions('DdeI');
                                my @cuts2 = $analysis -> positions('AflII');

                                my @cuts = (@cuts1, @cuts2);
                                my @sorted_cuts = sort {$b <=> $a} @cuts;

                                my @sizes;
                                for (my $i = 0; $i <= scalar @sorted_cuts; $i++)
                                {
                                        if ($i == 0)
                                        {
                                                my   $fragment   =   $seq   ->   length   -
$sorted_cuts[$i];
                                                push @sizes, $fragment;
                                        }
                                        else
                                        {
                                                my  $fragment  =  abs($sorted_cuts[$i]  -
$sorted_cuts[$i-1]);
                                                push @sizes, $fragment;
                                        }
                                }
                                my @sorted_sizes = sort {$b <=> $a} @sizes;
                                $apico_bands = join(", ", @sorted_sizes);


                                my %type = ("317, 107, 103, 69, 17, 14, 12" => "I", "167,
150, 107, 103, 69, 17, 14, 12" => "II", "317, 117, 107, 69, 17, 12" => "III");
                                my $allele = join(", ", @sorted_sizes);
                                if ($type{$allele} ne "")
                                {
                                        $apico = $type{$allele};
                                }
                                else
                                {
                                        $apico = "New allele";

                                        print NEWFILE "$prefix\n";
                                        print NEWFILE "Apico\t";
                                        print NEWFILE join(", ", @sorted_sizes)."\t";
                                        print NEWFILE "New allele\n\n";
                                }

                                print "$prefix\n";
                                print "SAG1\t$sag1_bands\t$sag1\n";
                                print "5-SAG2\t$sag2_5_bands\t$sag2_5\n";
                                print "3-SAG2\t$sag2_3_bands\t$sag2_3\n";
```

```
                        print "alt-SAG2\t$alt_sag2_bands\t$alt_sag2\n";
                        print "SAG3\t$sag3_bands\t$sag3\n";
                        print "BTUB\t$btub_bands\t$btub\n";
                        print "GRA6\t$gra6_bands\t$gra6\n";
                        print "c22-8\t$c22_8_bands\t$c22_8\n";
                        print "c29-2\t$c29_2_bands\t$c29_2\n";
                        print "L358\t$l358_bands\t$l358\n";
                        print "PK1\t$pk1_bands\t$pk1\n";
                        print "Apico\t$apico_bands\t$apico\n";
                        print "\n";

                        print                                                OUTFILE
"$prefix\t$sag1\t$sag2_5\t$sag2_3\t$alt_sag2\t$sag3\t$btub\t$gra6\t$c22_8\t$c29_2
\t$l358\t$pk1\t$apico\n";
                    }
                }
            }
}
close OUTFILE;
close NEWFILE;
```

2.5. REFERENCES

BOLGER, A. M.; LOHSE, M.; USADEL, B. Trimmomatic: A flexible trimmer for Illumina sequence data. **Bioinformatics**, v. 30, n. 15, p. 2114-2120, 2014.

CINGOLANI, P.; PLATTS, A.; WANG, L. L.; COON, M.; NGUYEN, T.; WANG, L.; LAND, S. J.; LU, X.; RUDEN, D. M. A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff. **Fly**, v. 6, n. 2, p. 80-92, 2012.

HOWE, D. K.; SIBLEY, L. D. *Toxoplasma gondii* comprises three clonal lineages: Correlation of parasite genotype with human disease. **The Journal of Infectious Diseases**, v. 172, n. 6, p. 1561-1566, 1995.

HUSON, D. H.; BRYANT, D. Application of phylogenetic networks in evolutionary studies. **Molecular Biology and Evolution**, v. 23, n. 2, p. 254-267, 2005.

LANGMEAD, B.; SALZBERG, S. L. Fast gapped-read alignment with Bowtie 2. **Nature Methods**, v. 9, n. 4, p. 357-359, 2012.

LEINONEN, R.; SUGAWARA, H.; SHUMWAY, M.; on behalf of the International Nucleotide Sequence Database Collaboration. The Sequence Read Archive. **Nucleic Acids Research**, v. 39, n. Suppl_1, p. D19-D21, 2010.

LI, H.; HANDSAKER, B.; WYSOKER, A.; FENNELL, T.; RUAN, J.; HOMER, N.; MARTH, G.; ABECASIS, G.; DURBIN, R.; SUBGROUP, G. P. D. P. The Sequence Alignment/Map format and SAMtools. **Bioinformatics**, v. 25, n. 16, p. 2078-2079, 2009.

MCKENNA, A.; HANNA, M.; BANKS, E.; SIVACHENKO, A.; CIBULSKIS, K.; KERNYTSKY, A.; GARIMELLA, K.; ALTSHULER, D.; GABRIEL, S.; DALY, M.; DEPRISTO, M. A. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. **Genome Research**, v. 20, n. 9, p. 1297-1303, 2010.

Oracle VM VirtualBox. Oracle Corporation, 2019. Disponível em: http://www.virtualbox.org/. Acesso em: 20 de agosto de 2018.

PENA, H. F. J.; GENNARI, S. M.; DUBEY, J. P.; SU, C. Population structure and mouse-virulence of *Toxoplasma gondii* in Brazil. **International Journal for Parasitology**, v. 38, n. 05, p. 561-569, 2008.

Picard Toolkit. Cambridge, MA: Broad Institute, GitHub Repository, 2019. Disponível em: http://broadinstitute.github.io/picard. Acesso em: 01 de abril de 2017.

RAJ, A.; STEPHENS, M.; PRITCHARD, J. K. FastSTRUCTURE: Variational inference of population structure in large SNP data sets. **Genetics**, v. 197, n. 2, p. 573-589, 2014.

SHWAB, E. K.; ZHU, X. Q.; MAJUMDAR, D.; PENA, H. F. J.; GENNARI, S. M.; DUBEY, J. P.; SU, C. Geographical patterns of *Toxoplasma gondii* genetic diversity revealed by multilocus PCR-RFLP genotyping. **Parasitology**, v. 141, n. 4, p. 453-461, 2014.

SU, C.; ZHANG, X.; DUBEY, J. P. Genotyping of *Toxoplasma gondii* by multilocus PCR-RFLP markers: A high resolution and simple method for identification of parasites. **International Journal for Parasitology**, v. 36, n. 7, p. 841-848, 2006.

WALKER, B. J.; ABEEL, T.; SHEA, T.; PRIEST, M.; ABOUELLIEL, A.; SAKTHIKUMAR, S.; CUOMO, C. A.; ZENG, Q.; WORTMAN, J.; YOUNG, S. K.; EARL, A. M. Pilon: An integrated tool for comprehensive microbial variant detection and genome assembly improvement. **PLoS ONE**, v. 9, n. 11, p. 1-14, 2014.

3. INTEGRATE MULTILOCUS PCR-RFLP GENOTYPING, MULTILOCUS SEQUENCE TYPING (MLST) WITH WHOLE GENOME SEQUENCING TYPING (WGST) IN *T. gondii*

ABSTRACT

Multilocus polymerase chain reaction-restriction fragment length polymorphism (PCR-RFLP) and multilocus sequence typing (MLST) have been widely used to genotype microorganisms. Accumulation of data generated by these methods has helped to reveal genetic diversity, population structure and transmission of many microbial pathogens. Recent advances in DNA sequencing technologies have made it possible to identify microorganisms by whole genome sequence typing (WGST) to reach the highest resolution possible. While WGST is replacing PCR-RFLP and MLST for genotyping, invaluable database generated by the latter may not be easily related to the new system. To alleviate this problem, we made effort to develop a protocol that can generate PCR-RFLP and MLST data from WGST results, therefore make it easier to integrate data generated by different systems. Here, we used the protozoan parasite *Toxoplasma gondii* as a system to show that such an integration will bridge different methods and enhance the future usage of existing databases.

Keywords: Multilocus polymerase chain reaction-restriction fragment length polymorphism. Multilocus sequence typing. Whole genome sequence typing. *Toxoplasma gondii*

3.1. INTRODUCTION

Toxoplasma gondii is a ubiquitous protozoan parasite that infects mammals and birds. About one third human population is chronically infected (MONTOYA; LIESENFELD, 2004; DUBEY, 2010). Though majority of infection in humans are asymptomatic, in some individuals the infection causes ocular toxoplasmosis which may affect vision. In pregnant women, primary infection may cause damages in developing fetuses, leading to hypoencephaly, calcification of cranial and still birth. In AIDS patients, reactivation of *T. gondii* can cause acute encephalitis which is often fetal (MONTOYA; LIESENFELD, 2004).

Molecular typing methods are important tools to identify individual pathogens with relatively high resolution. Among many such methods, multilocus PCR-RFLP and microsatellite markers as well as MLST have been often used to distinguish different strains of *T. gondii* worldwide (LEHMANN et al., 2006; KHAN et al., 2007; AJZENBERG et al., 2010; SU et al., 2010). In the past two decades, accumulating of typing data based on samples collected worldwide allowed us to reveal genetic diversity, population structure and transmission pattern of *T. gondii* (SU et al., 2012; SHWAB et al., 2014). Recent analysis of multilocus PCR-RFLP data of *T. gondii* samples from North America revealed partition of different genotypes among domestic and wild animals and along the spatial gradient from human settlement to wild environment, suggesting human impact of *T. gondii* transmission (JIANG et al., 2018).

The multilocus PCR-RFLP markers are easy to use and have a high resolution in identifying *T. gondii* isolates (SU et al., 2006). These markers were originally

developed, based on DNA sequence polymorphisms, where clonal lineages are classified in type I, II and III and non-clonal alleles are denoted as u-1 and u-2.

For epidemiological studies and surveillance of pathogen transmission, it is important to establish a standardized typing method and apply to a large number of samples for a long period of time (PÉREZ-LOSADA et al., 2013). Along with rapid advance of DNA sequencing technologies, WGST is becoming the preferred method due to the highest resolution it can achieve (PÉREZ-LOSADA et al., 2013). However, this new form of typing data may not directly retrospectively fit to the databased previously established by PCR-RFLP and MLST. This is particularly true for *T. gondii*, in which a large number of samples were typed by these methods using DNA samples that contained significant amount of host DNA, which cannot be used to generate WGST data, as the latter will need highly purified *T. gondii* genomic DNA for sequencing. To alleviate this problem, we developed a protocol to generate multilocus PCR-RFLP and MLST data from WGST sequences, therefore keeping existing database useful in the era of WGST genotyping.

3.2. MATERIAL AND METHODS

3.2.1. Generate PCR-RFLP genotypes from whole genome sequence data

For new *T. gondii* samples that have not been genotyped by multilocus PCR-RFLP markers but the whole genome sequences are available, there are three steps to generate genotype profile from the genome sequence data (Figure 11). Step 1 is to obtain PCR-RFLP marker sequences using the Bioinformatics Virtual Machine

package. Step 2 is to digest sequences in silico by corresponding restriction enzyme(s). Step 3 is to obtain genotyping data using a PCR-RFLP formula.

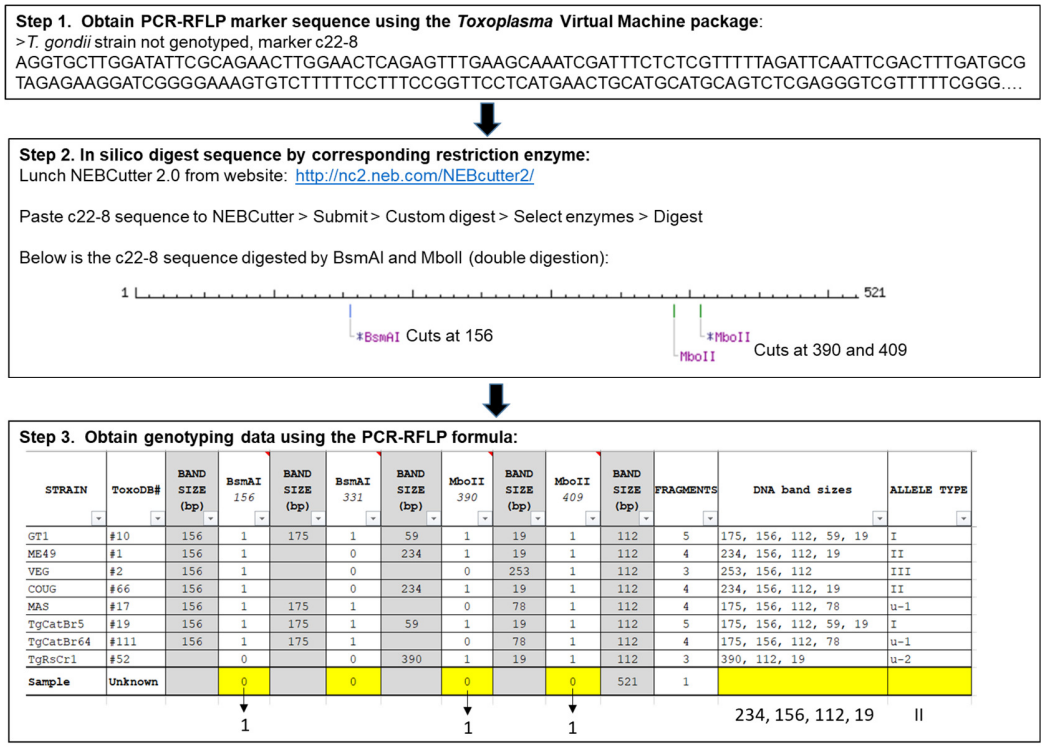


Figure 11: Steps to generate genotype profile from the genome sequence data

Step 1: Obtain PCR-RFLP marker sequences using the Toxoplasma Virtual Machine package

In a chapter 2, we described the pipeline to map *T. gondii* whole genome Illumina short reads to the reference genome of ME49. The mapped short reads can be used to assemble PCR-RFLP marker sequences for any sequenced *T. gondii* strains. To accomplish this, DNA sequences of the 10 PCR-RFLP markers are obtained for ME49 from previously publish report (SU et al., 2010). These sequences are summarized in Supplementary Data (PCR-RFLP ME49 sequences) in this report. Each of these sequences is aligned to ToxoDB (http://toxodb.org/) reference genome

of ME49 by BLAST to get the coordinates, which defines the exact location of match. The coordinates are summarized in Table 2. Then, the coordinates are used to assemble marker sequences for different samples, using a Pilon software, version 1.22 (WALKER et al., 2014). For detailed procedure, see chapter 2.

Marker	Size	Chromosome	Initial position	Final position
SAG1	390 bp	TGME49_chrVIII	2664322	2664711
5'-SAG2	242 bp	TGME49_chrVIII	4757968	4758209
3'-SAG2	222 bp	TGME49_chrVIII	4759062	4759283
alt. SAG2	546 bp	TGME49_chrVIII	4757992	4758537
SAG3	226 bp	TGME49_chrXII	510601	510826
BTUB	411 bp	TGME49_chrIX	870907	871317
GRA6	344 bp	TGME49_chrX	7272896	7273239
c22-8	521 bp	TGME49_chrIb	1854996	1855516
c29-2	446 bp	TGME49_chrIII	662796	663241
L358	417 bp	TGME49_chrV	2241407	2241823
PK1	903 bp	TGME49_chrVI	2682178	2683080
Apico	639 bp	KE138841	7590	8228

Table 2: Coordinates of ME49 sequences in ToxoDB.

Step 2: Digest sequence in silico by corresponding restriction enzyme(s)

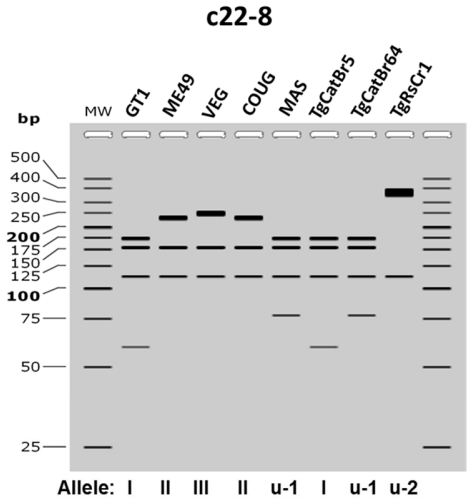
In this step, the assembled sample sequences are treated with restriction enzymes in silico (Figure 11). First, launch NEBcutter 2.0 from website http://nc2.neb.com/NEBcutter2 (VINCZE; POSFAI; ROBERTS, 2003), then paste each sample sequence to NEBcutter, select corresponding restriction enzymes to digest the sequence in silico, and record cutting sites for the enzymes. In Figure 11 (Step 2), the c22-8 sequence of a sample is double-digested with enzymes BsmAI and

MboII. The BsmAI cuts the sequence at position 156, and MboII cuts at positions 390 and 409, respectively.

Step 3: Obtain genotyping data using the PCR-RFLP formula

In this step, we established a PCR-RFLP formula for each marker that can be used to generate DNA fragment banding pattern. These formulas are provided in CD a part attached this report. In these formulas, the absence of restriction site is coded as 0, the presence of a restriction site is coded as 1. DNA sequences from eight *T. gondii* reference strains are used to generate corresponding fragments for comparison in Supplementary Data (Gel simulation of 8 reference strains). Figure 12 shows gel simulation for marker c22-8. The corresponding cutting sites for enzymes BsmAI and MboII are coded as 1s, which produce DNA fragment pattern matching to allele type II (Figure 11- Step 3).

A.



B.

Strain	ToxoDB genotype	DNA fragment sizes	Allele type
GT1, RH	#10	175, 156, 112, 59, 19	I
ME49, PTG	#1	234, 156, 112, 19	II
VEG, CTG	#2	253, 156, 112	III
COUG (TgCgCa1)	#66	234, 156, 112, 19	II
MAS	#17	175, 156, 112, 78	u-1
TgCatBr5	#19	175, 156, 112, 59, 19	I
TgCatBr64	#111	175, 156, 112, 78	u-1
TgRsCr1	#52	390, 112, 19	u-2

Figure 12: Gel simulation of marker c22-8 by SnapGene (GSL Biotech LLC, USA)

3.2.2. Retrieve sequences of MLST markers from whole genome sequence alignment

This step is similar to retrieving sequences for the PCR-RFLP markers. The MLST DNA sequences of the reference strain ME49 are obtained for the 8 intron markers (KHAN et al., 2007). See Supplementary Data (MLST ME49 sequences). These sequences were BLAST to ToxoDB (http://toxodb.org/) genomic sequence of ME49 to get their coordinates (Supplementary Data - MLST ME49 Coordinates). Then, the coordinates are used to assemble the 8 intron sequences for different samples. For detailed procedure, see chapter 2 of this report. DNA sequences

corresponding to the 8 MLST intron markers of the 62 sequenced strains are generated.

3.3. RESULTS

All PCR-RFLP markers are processed for sequenced *T. gondii* isolates, the genotyping results are generated and Table 3 shows the results for 8 reference strains and 10 additional samples. The patterns match the patterns observed in the gel simulation. The phylogenetic network based on the 8 MLST intron sequences of the 62 sequenced strains is presented in Figure 13.

Strain	#ToxoDB	SAG1	3'5' SAG2	alt. SAG2	SAG3	BTUB	GRA6	c22-8	c29-2	L358	PK1	Apico
GT1	#10	I	I	I	I	I	I	I	I	I	I	I
ME49	#1	II or III	II	II	II	II	II	II	II	II	II	II
VEG	#3	II or III	III	III	III	III	III	III	III	III	III	III
COUG (TgCgCa1)	#66	I	II	II	III	II	II	II	u-1	I	u-2	I
MAS	#17	u-1	I	II	III	III	III	u-1	I	I	III	I
TgCtBr5	#19	I	III	III	III	III	III	I	I	I	u-1	I
TgCtBr64	#111	I	I	u-1	III	III	III	u-1	I	III	III	I
TgRsCr1	#52	u-1	I	II	III	I	III	u-2	I	I	III	I
ARI	#5	u-1	II	II	II	II	II	II	II	I	II	I
CAST	#28	I	I	I	I	I	I	II	I	III	I	III
TgCtCo5	#61	I	I	II	III	I	III	II	I	I	u-2	I
FOU	#6	I	I	I	III	I	II	u-1	I	I	I	I
TgA105004	#203	I	I	I	III	I	II	II	III	III	I	III
p89	#8	I	III	III	III	III	III	II	III	III	III	III
RUB	#98	I	I	II	I	III	III	III	III	I	III	III
TgCtBr9	#42	I	I	I	III	III	II	I	I	I	u-1	I
TgCtPRC2	#9	u-1	II	II	III	III	II	II	III	II	II	I
VAND	#60	I	I	II	I	III	III	III	I	III	III	I

Table 3: Genotype summary for 8 reference strains and additional 10 samples.

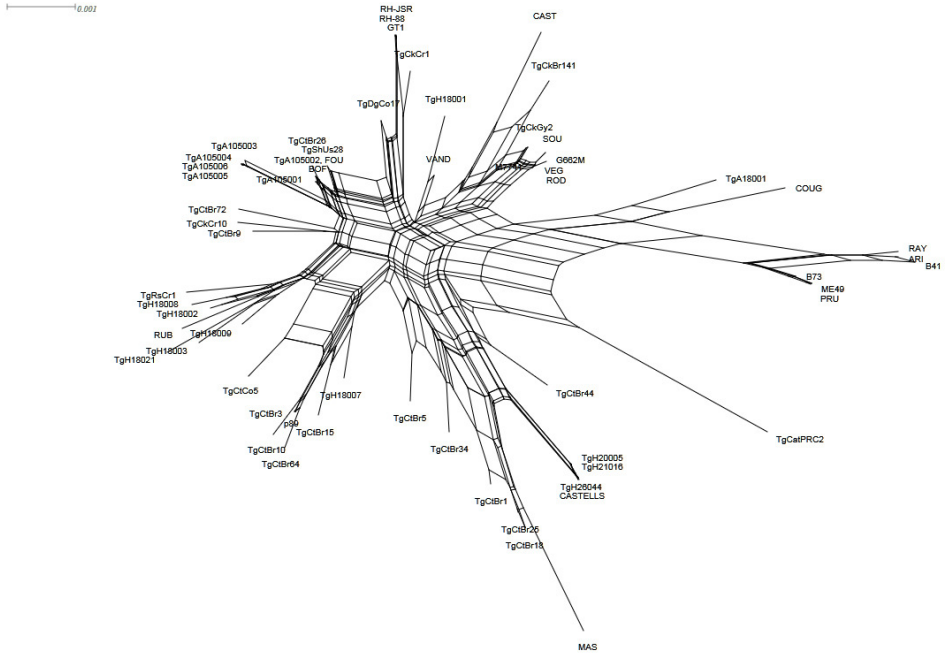


Figure 13: Phylogenetic network of 8 MLST intron sequences of 62 *T. gondii* strains

3.4. DISCUSSION

In this study we showed that the whole genome sequence typing data can be used to generate PCR-RFLP allele types or the MLST data, which can retrospectively integrate to existing database. This is particularly useful for pathogens that could not be revived or cultured to generate WGST data. Using *T. gondii* as an example, it is shown that this is a simple procedure that can be easily implemented. The PCR-RFLP profiles generated from WGST data matched with the results from the conventional PCR-RFLP method, which suggest the high quality of the typing data deposited in ToxoDB database.

The use of this method can be very useful to predict possible results for PCR-RFLP and MLST genotyping in a short time, which can even be confirmed later by

the conventional agarose gel method. With the advent and refinement of sequencing techniques, the use of tools such as the one proposed in this work tend to help methods that are currently used, which are more time consuming.

Another advantage of this tool is the possibility of extracting other regions of the DNA and testing them in already sequenced isolates, so that possible new markers could be tested quickly, avoiding the time of preparation of agarose gel and PCR-RFLP and MLST procedures.

3.5. SUPPLEMENTARY MATERIAL

3.5.1. PCR-RFLP ME49 sequences

>SAG1-ME49
CAATGTGCACCTGTAGGAAGCTGTAGTCACTGCTGATTCTCGCTGTTCTCGGCAAGGGCTGACGACCGGAGTACAGTTTTT
GTGGGCAGAGCCGCTGTGCAGCTTTCCGTTGTTCTCGGTTGTGTCACATGTGTCATTGTCGTGTAAACACACGGTTGTATG
TCGGTTTCGCTGCACCACTTCATTATTTCTTCTGGTTTTTTGGCGAGTATGTTTCCGAAGGCAGTGAGACGCGCCGTCACG
GCAGGGGTGTTTGCCGCGCCCACACTGATGTCGTTCTTGCGATGTGGCGCTATGGCATCGGATCCCCCTCTTGTTGCCAAT
CAAGTTGTCACCTGCCCAGATAAAAAATCGACAGCCGCGGTCATTCTCACACCGACGGAGAACCAC

>5-SAG2-ME49
GAAATGTTTCAGGTTGCTGCAGTGACCCATCTGCGAAGAAAACGAGCGCTGCTTGCGATTCTGTGTGTTGTTTCACGGGTT
GCAGTTCTAGGAACTGAGTTGTGATTGTGCACAATTGCGGTGTGACACCTTCTGTCTCGTTCCAATCTTTGTCTTGTCGGA
ACTATGAGTTTCTCAAAGACCACGAGCCTAGCGTCGCTAGCGCTCACGGGCTTGTTTGTTGTGTTCAAGTTCGCTCTTGC

>3-SAG2-ME49
AACGTTTCACGAAGGCACACCCGCCGGAACACTGGTTGTGTCTGGCGGAAAAGCGCAGTACTGCAGCAGTGTGTCGGCAAA
CAAGTTCCTCTGTCACATAATCGCAGATCCGGAAATACATTTTCTCGCGTTCCTCCCAGGATCTACGCCAAGAACAATCGC
AATCCCACCCCTCAACAACGTTTTCTTTGCACGACAATTTCGAAGCGGAGGCATGAAAAT

>alt-SAG2-ME49
ATTTCGACCAGCGGGAGCACCTGCTTCGGCAACGCACTTGTAACAGAGAACCTGAGGTTTCTCGGGAGTACCGTCGTAAGA
CAGTGTGTAGGTAGCAGGACCTTTCGGGGGCTGCTCGACCTTAGCTTTCAAGACCGCACCTGGAAGGACAGTCGTCAACTT

CCTCGAGTCTGTGCATTCCTTGCCATAAAAGACATCACCTTCGCCACTGGGACTGATGGTTAGTTTATCCCCACATTGGAA
GACAACGGAACCACTGGAGGGTGCCTCAACAGTCTTCGTTGCGCCGGCAGTGCACTCAATGGGCGCTGGCGTCTCGGTGGT
GGACGCAAGAGCGAACTTGAACACAACAAACAAGCCCGTGAGCGCTAGCGACGCTAGGCTCGTGGTCTTTGAGAAACTCAT
AGTTCCGACAAGACAAAGATTGGAACGAGACAGAAGGTGTCACACCGCAATTGTGCACAATCACAACTCAGTTCCTAGAAC
TGCAACCCGTGAAACAACACACAGAATCGCAAGCAGCGCTCGTTTTCTTCGCAGATGGGT

>SAG3-ME49
CACAAGGAGACCGAGAAGGAACGCCAGGGCAACGGAAGCCATTCGCGAACTCGCGCCGGCTGAACTTCCTGTTCCAGCTTG
AGTTTCCCCGTCAGTTCCTGGCTGCAATCCTCCGCTTCCTGGATGGCCGCCTCCGCCCCTCTCCCACTTTCTGGGGTTCCC
AGCAACTCTCACTTTGACGTTGCAGAAGGGCGGCCCGTCCACAGTGAGTGAACACCCGACAAGA

>BTUB-ME49
GAGGTCATCTCGGACGAACACGGCATTGATCCGGTGAGGGAGAACGGGTTCCGGAGTGCGGGTTCCCGGACGCTGTCTTTT
GTCCTGTATTCTGCGTGACAGCTTCGCCAGTGTAAATGGAGCCGTTTTCCCTGTGGAGAATCGCGGAGAATGGAGGACTTT
TCCGAGCTCGCAGGTGCCACCCTCCACCTCGACGCAACCAAGTGCACGCACATTTGCCGGGTGGTGACCCTAAGACACCGC
AGGTCTACCTGCGCGTTTTTTTCAGTCTTTGCACACAGTTGCACCGAAAGTCATGTTTTTTGCGAAGACATCGGTCGTTCT
GGTGGGGGAATACTTCACTCCTGTTCGCGCCTATGTGCGCAGACAGGTGTCCACCCTCCGCATACGGGCGTCCGGGTGTTC
CTACAA

>GRA6-ME49
TCGCCGAAGAGTTGACATAGTCTTCAGTGGTCCCCACTGCTTCTTGCTGTCCACCGCTCGAACCGGTTTCCGAAGGGGTCT
GCCTAACACCACCGCTGTCTGCTGCGACAGCGACTCCACCCAACGAATTGACGAGTACACCCATGAAGACTACAAAGACCA
CAGCAACTGTGGAGACAGTTAAAGGACAGAAGTTACGCTTCTGCCTCAGATAGATGCCACCGTGTGCCATTTCGCCGACAC
TCCCAAGAAAAAAAGGTAGTCGCCAACAAGCACGCTACGTGGCACCACGAAGCTGTTTTCCTGCTGTTTCAAAAAAGCGAC
CCAGGTCACCTGCTCGGAAA

>c22-8-ME49
AGGTGCTTGGATATTCGCAGAACTTGGAACTCAGAGTTTGAAGCAAATCGATTTCTCTCGTTTTTAGATTCAATTCGACTT
TGATGCGTAGAGAAGGATCGGGGAAAGTGTCTTTTTCCTTTCCGGTTCCTCATGAACTGCATGCATGCAGTCTCGAGGGTC
GTTTTTCGGGGTAGTGCTCGTCGCTACAGCTGTGCGAGTTCACAGGTTTGCTGCATTTTCCGTTCAGCGAAAATGCTGAAA
AGTTCCAGTGCGTTTCCAGTGTTGTGGGAGGCAGCGGTTGCCTTCCAGTGGCACCTCCGAATTTGCTTCTCGAGGAATTTC
AATCTCTCCAAAACCCTGCACAAAGAAGCGAAGGGATCTGAGAGAAAGGCAGGGGAGATGACAGAAGGTTCGCTCTTCTTG
ACTTCGGAAATTCTTCGATTGCTTCACGGCGGGAATGCGAACCGCATGCCGTCCGTTTTTGAAAAAGAAACGAAACGCAGC
GCGAAAGGAACGAGAAAGGCGTCCACGTAGAGAGA

>c29-2-ME49
AGTTCTGCAGAGTGTCGCTTTTTCCGGAAGCGGAGCATGTGGACACGTATGCCATTTCAAATGAATCGAGGACCGGAGGGC
TAACGAGACGCCAGTGTACACACTGGAGGAGGTCCGTCCATTCCTGTTCCTCGTTTTCCGATTCTGTGTGTCTTTCCCATT
TTCGCGCTCTTCCACTTCTATGCGTTTATTTGATTGCCTTCCAGCCATGCTGTTCAACAAGAGTGGCTTTTCGCGGGAGTA

AGAACTCAGCCACATGCAGTACACAAAAGTCCCTCGAATGCTCTCTGAACACACCGAACTCCCGGGAAGGTGCTCACAAAC
ATGCCCACCACCGGATATCCCGTTGTGCTAAAGTGTTTCATCAACTTCTCTCTCTTGATATCCATTCTCACATCACTCCTA
CGTATTCTGCTCAGTCCGAACAGGCGCCTCTCTCCTAGACA

>L358-ME49

CCCTCTGGCTGCAGTGCTGCGTCTTGGCCCTGCAAGGGTGCGAGTCTCTGCTCTTCCGGACGACACACTCGCATGCATGTC
CTCTTTCTGCCTTCGGCTTGTCGGATGCGTCCTGTCGGGGGAACGGAGCCTCTTCGTCTTCCCCGCTTCCTTCGCTCGGCT
GCGCGTCGACGCTGCAATGTCGCGCGGCCACCCCGCTGATAGACAAGAAGGCGAGGAGAGAGGCAGAGATCAGTTTCTCCT
GCATTCGCCTGCCGCCTCTCTCGCGACTCGTTTCGGACGGAGAAGCGGGGAGAGAAGCGAGCGACGAGACAGCGACGGAAA
CCGCGCAGCAGGAAGGTTTCGCTTTCTCCGCCCGTGTCGAGGAAGAGAACATTCCATTCTTCGCGGCATGCTGCGACTTGC
GCTACGCCTCCT

>PK1-ME49

TCATCGCTGAATCTCATTGCACTCCAAACTCCAAGAAGTGAACCCTGCGTCTCGAATAGGATGCATTAGTGGGTTTCCCTG
CGAGAGTGTTACTGCGATGCGAGTGGCAACGTGTTCGGAGCGCAATACCTTATGTGTAGGACAGTTAAGATGCAGAATTTG
TATGGGTGTCGTTCTTCCCGCATTTCGCTGAATAATTAACTGCGCAACATAATCCCAACAGTGTCACATGCTGACCCGAGT
ATCAGGGGCTTTTTTGTTGCCCATATGCTTTTCATAAGCAGTAGGGCACAATGGAAGACGATTTTGTCCAGACTGTTTCCG
CACCTGAATTTGCCACGTGCAGGTCATGCACAATTTTCTCTAGTGTAAACCAAAGTACACCTGTATTAGGTCTGAAAGAGA
GAGTTTTGCTAGTTGAGATCTATAGCCAACTTGTATAAGGCGATCCACATTGAACTGCGTTAAGGTGCAATCTGTACGGAA
TACAGGGATCGAATGTTTGGTGGCCTGGTACCAGTTTCGCGTTCGTTGGGCACACGTTTTCGCCGTGATGCAAAACCATGG
GTTTCCTTAAAACTCTCTCTTTTTGTGTCATCGACAACAAAGTGGCGAAATGCGTGCACTTTGTCACCACCAGGAGAGATC
GTTGCAATCAAGCTTATAAACAAAAGATCATTCAACGAGATCACCGATGCTGACCGCGTTTTCGTCGAGATCCAGGTAACG
TACTGACCACTCTACGAGACAGCTTGTTCTGGAGAACGGTTCTTCACGTACTGATGAACATACGTCACATGCACTCGTTAT
TCGCCATTTGGCAGCAGATGATTAGAGTCGCTGCCGACACATTCCAAACAGTTTTTTACTCCATGAAGAAGAGACTGATTG
TCTCCCTTTGCG

>Apico-ME49

TGCAAATTCTTGAATTCTCAGTTCAAATCTGAGTATCATTTTTTATAAAAAGGATGTGGTGAAATTTGGTAAACACAGCGG
ACTTTATAATAATTTAAATTTAGAGATAAATAGTTTTTATCTTTATTATTATCGATAAACTATAAAAAAATTATATAAGAA
TTTTTGTATTATTAAACTTAGATTTATTAATCAAAAGACTCTAAAAATAAATTACTTAATATTAATATAGAGTTTATAAAA
AAGTTAACTAAATTAATTATAAATATATGAAAATCCGTAAACTTTAAAAAAGTTTTGAGGGTTCGAGTCCCTTCTTCCTTA
TTTTTTTTAATATACTTGTGGCTGAGTGGGCAAAAGCAGTGAGCTCATAACTCATATAAAACGAAAGTTCGAATCTTTTCA
AGTATATAATAAAATAAATATAAATAAATTAATTTATGGATTGATGTCTGAGTGGTCAATAGAAATAGACTGTAAATCTAT
AGAAGTTATTCTTCATCGGTTCAAATCCGATTCAATTCATTAAAATTTAAATTATAAAATTATAATTTACTTAAGAAAAGT
GGCTGAGAGGTTTAAAGCACCAATTTGCTAGTTTGGTATATAATTAAATTTTATATCAAGGGTTCGAATCCC

3.5.2. Gel simulation of 8 reference strains

Figure S1: Gel simulation of 8 reference strains for SAG1 marker.

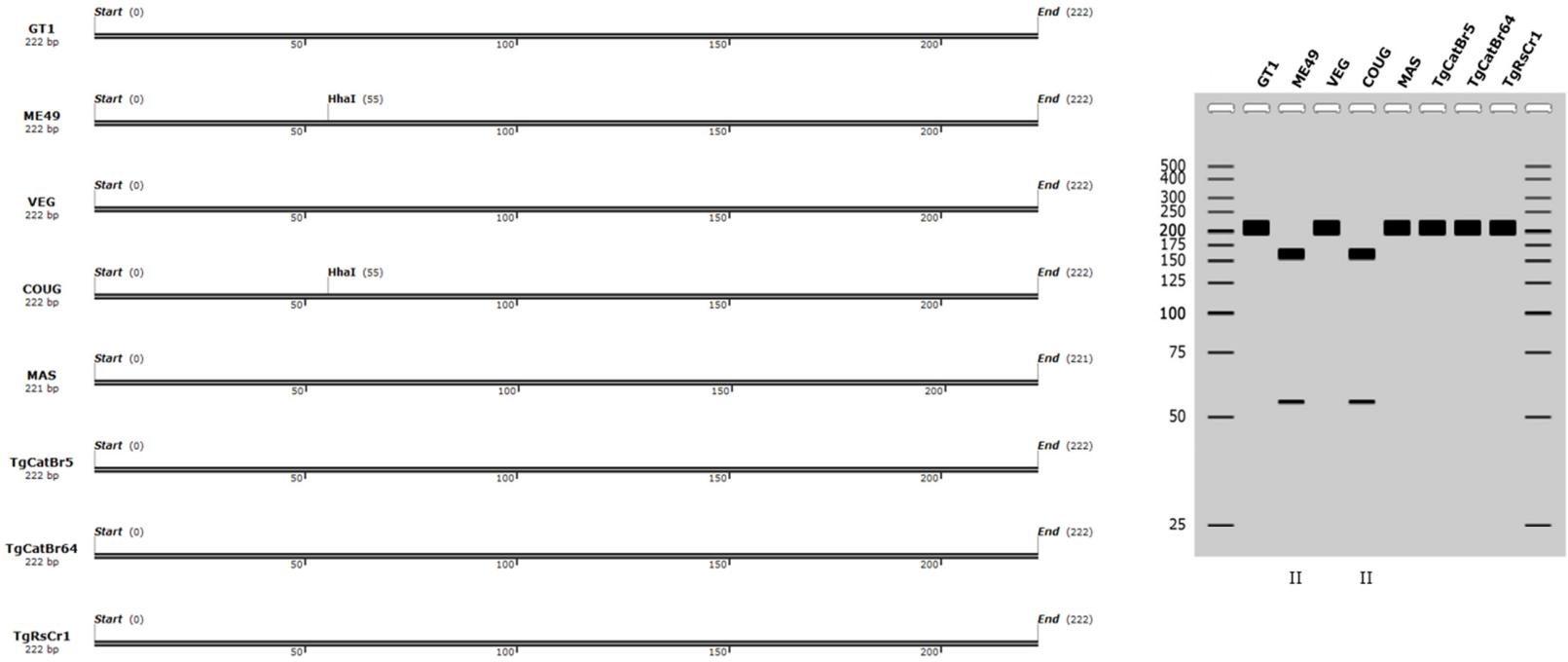


Figure S2: Gel simulation of 8 reference strains for 3'-SAG2 marker.

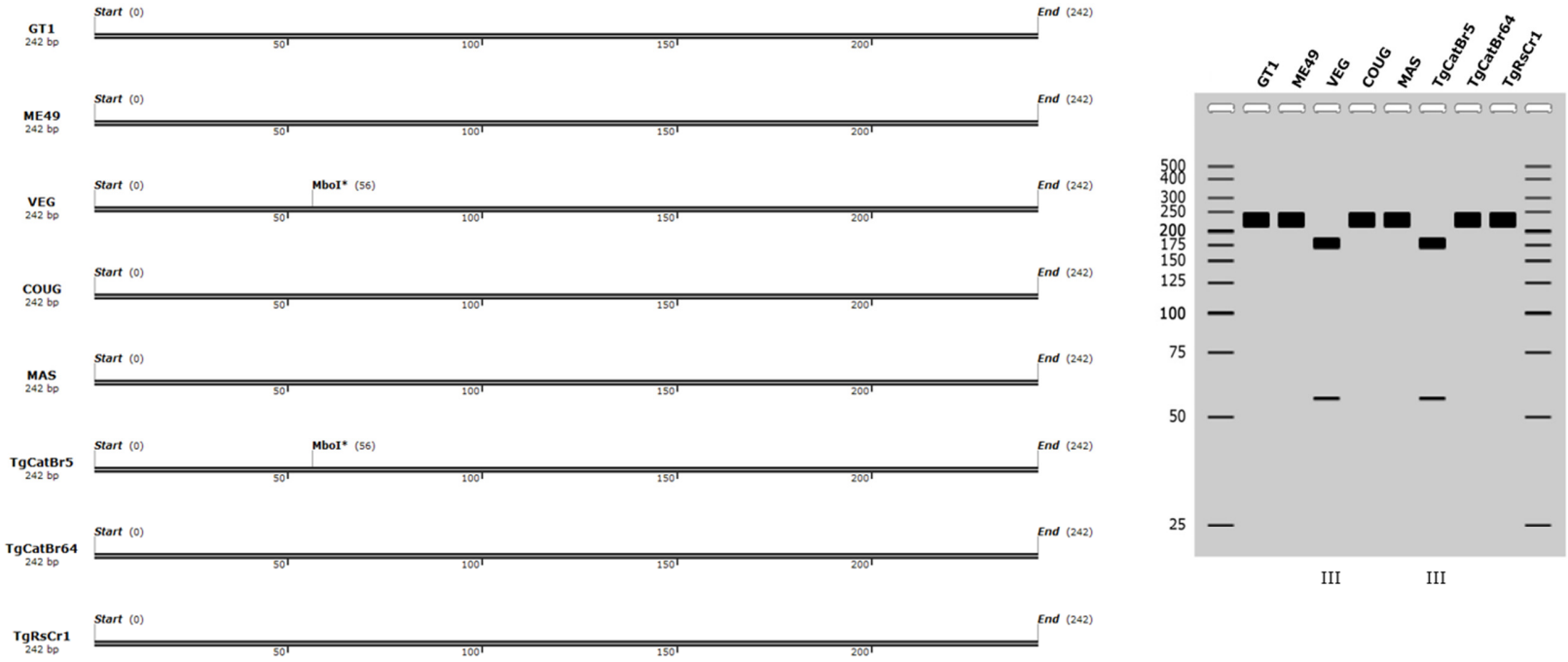


Figure S3: Gel simulation of 8 reference strains for 5'-SAG2 marker.

Figure S4: Gel simulation of 8 reference strains for alt. SAG2 marker.

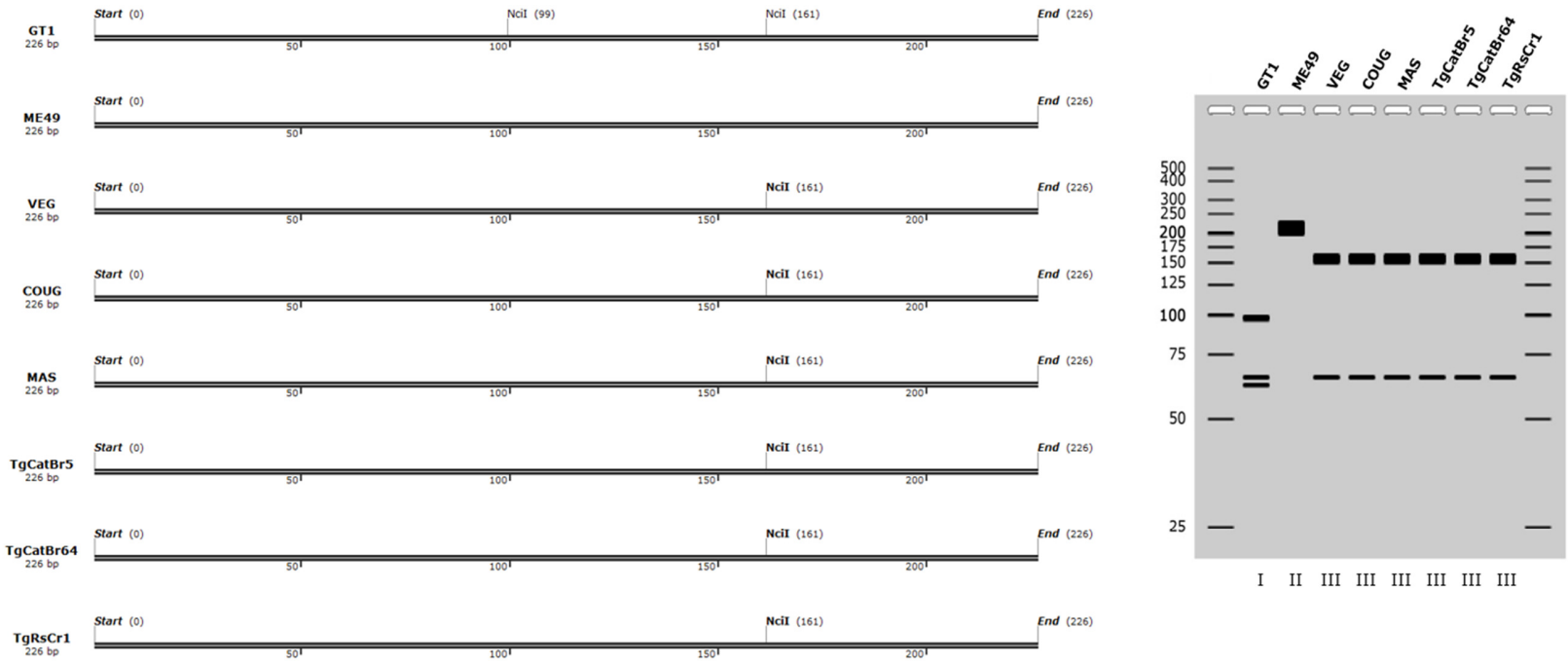


Figure S5: Gel simulation of 8 reference strains for SAG3 marker.

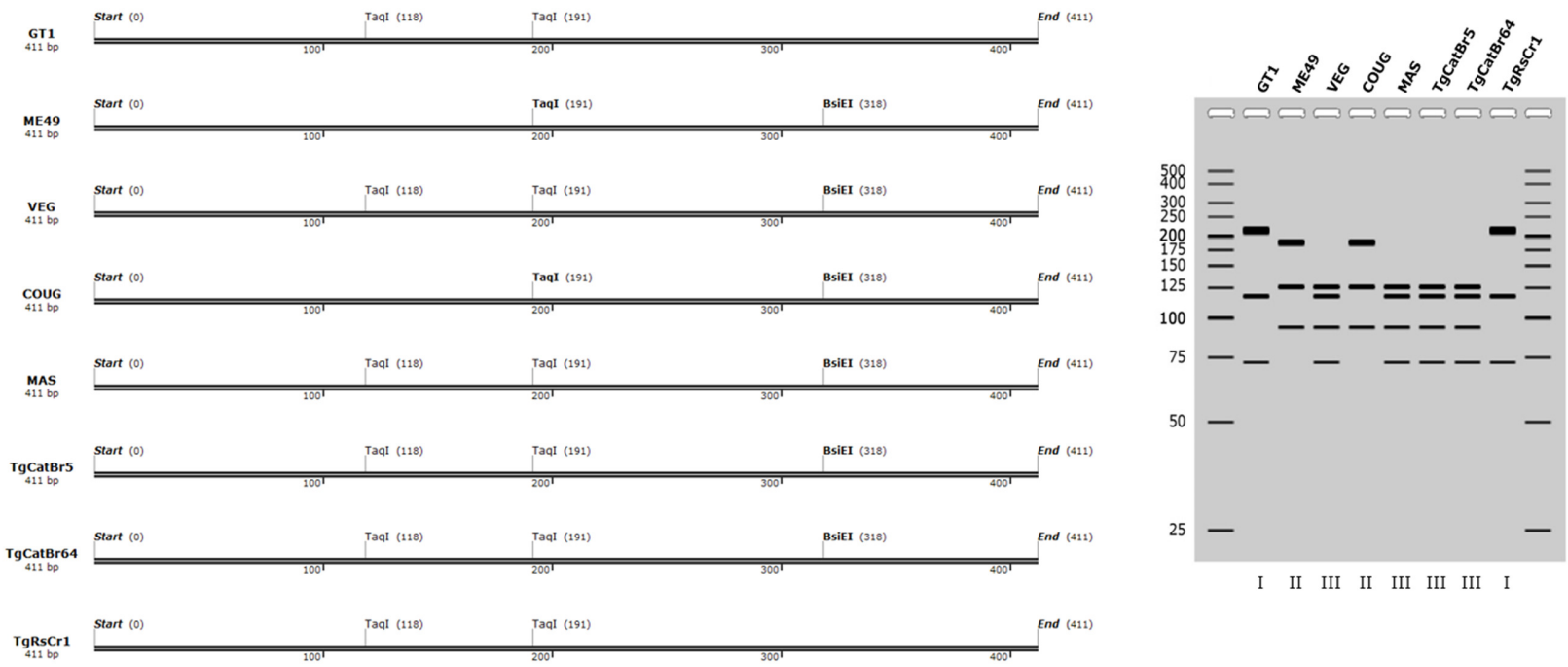


Figure S6: Gel simulation of 8 reference strains for BTUB marker.

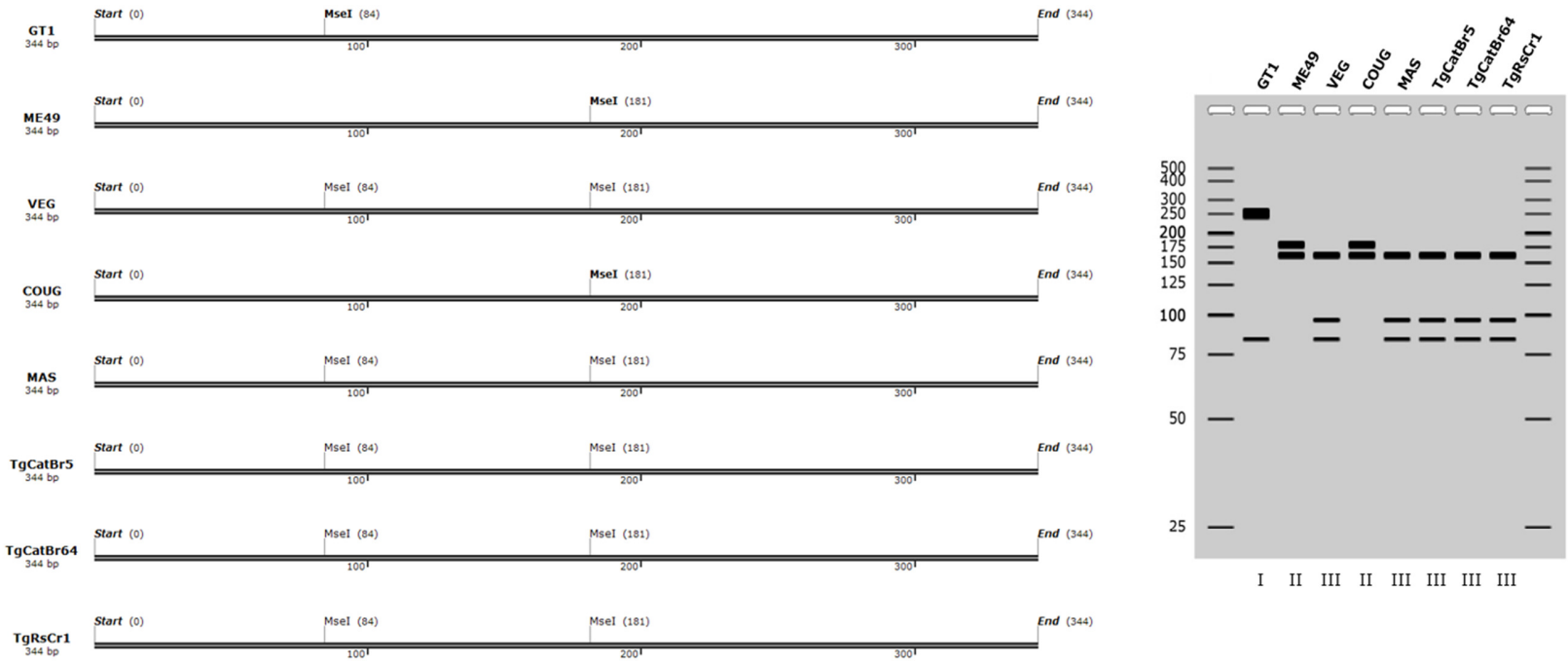


Figure S7: Gel simulation of 8 reference strains for GRA6 marker.

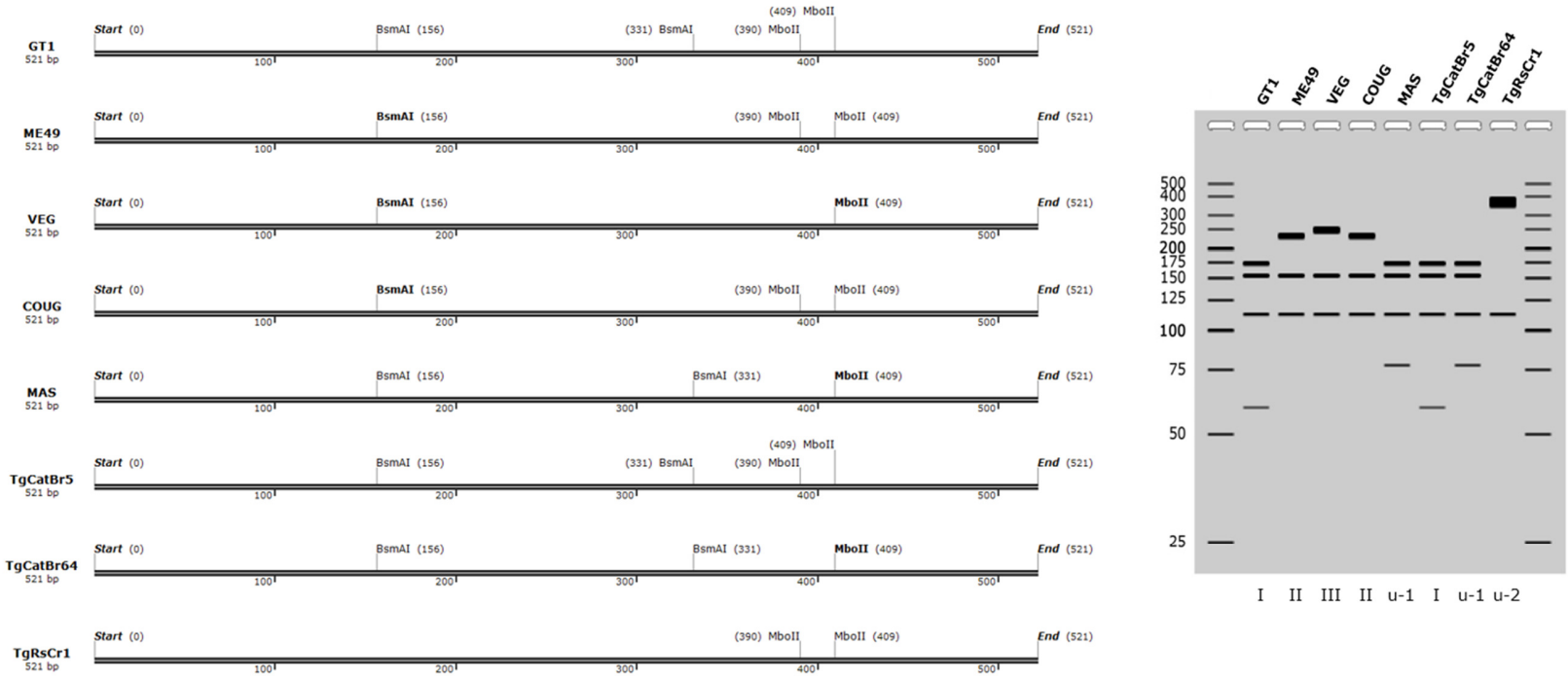


Figure S8: Gel simulation of 8 reference strains for c22-8 marker.

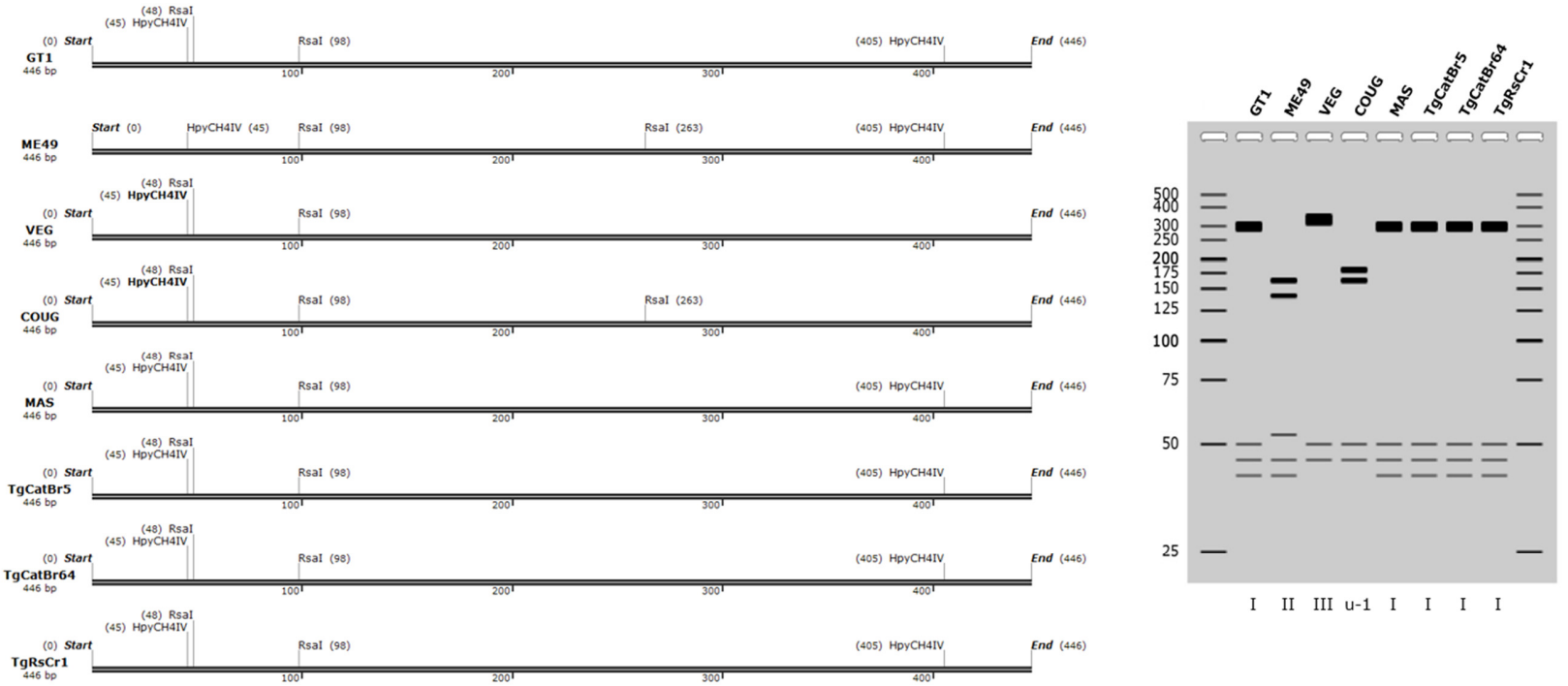


Figure S9: Gel simulation of 8 reference strains for c29-2 marker.

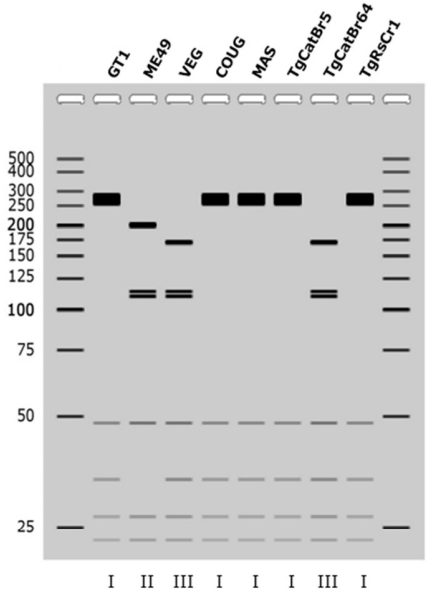


Figure S10: Gel simulation of 8 reference strains for L358 marker.

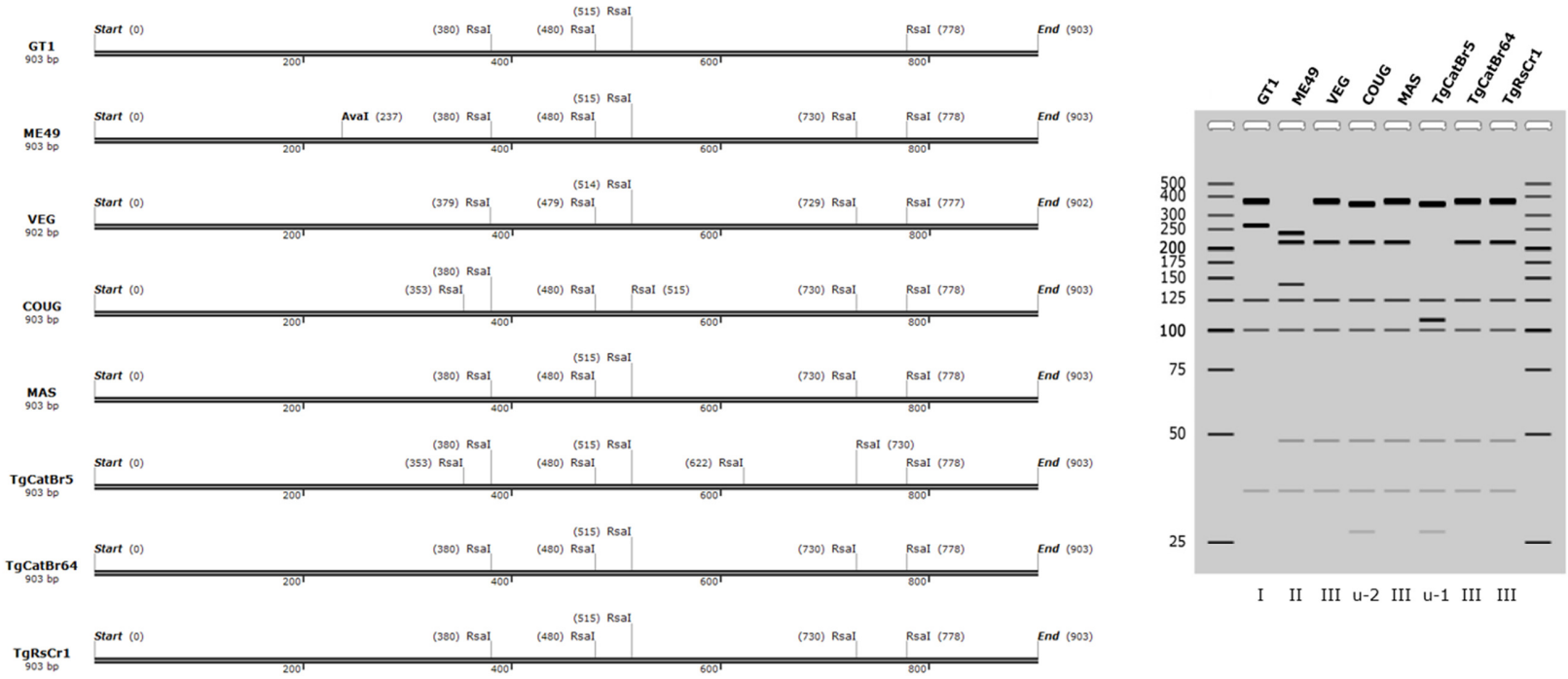


Figure S11: Gel simulation of 8 reference strains for PK1 marker.

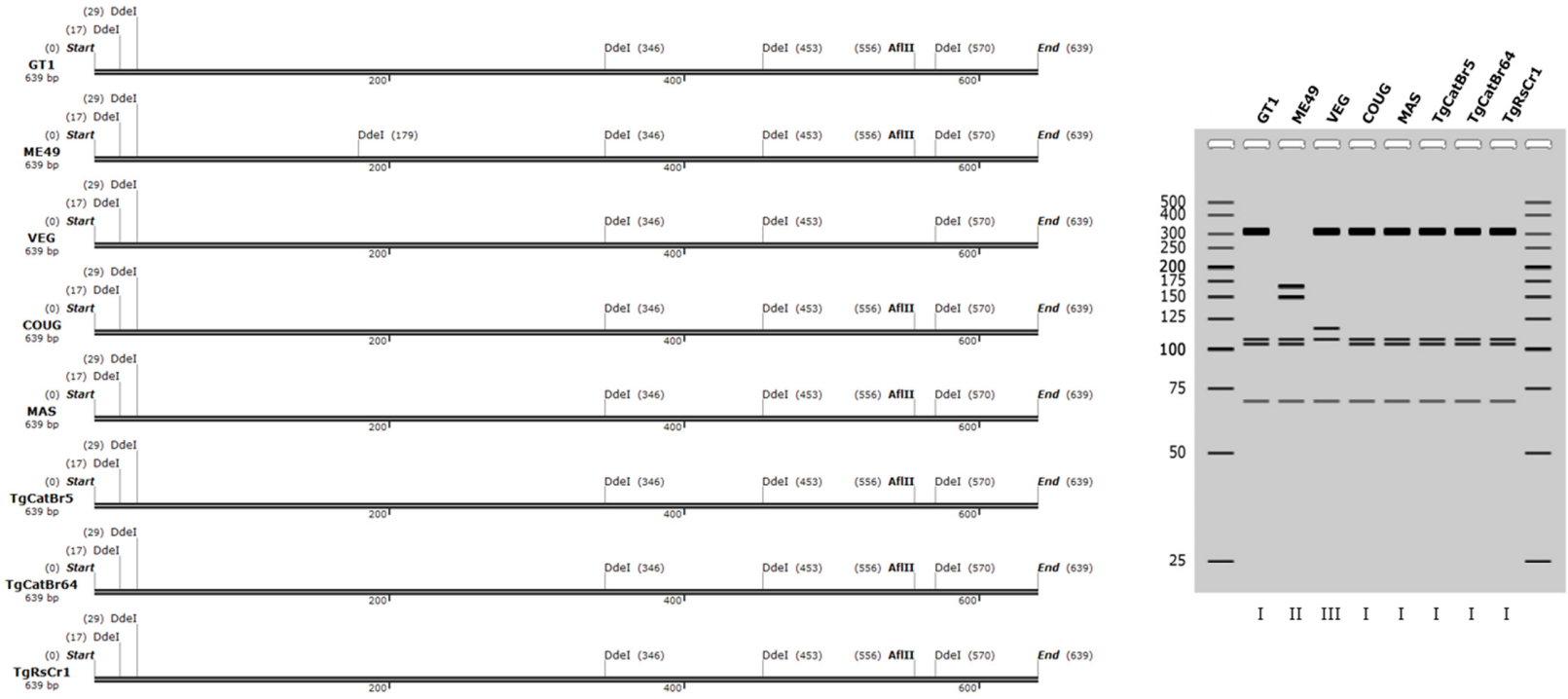


Figure S12: Gel simulation of 8 reference strains for Apico marker.

3.5.3. MLST ME49 sequences

>UPRT1, 466bp, AY143142.1 *Toxoplasma gondii* strain B7 type II uracil phosphoribosyl transferase (UPRT) gene, intron 1

GTAATCCTTCAACCGAAGTTTGCTTTCCGTGACTCTGCCTGTTGGTTATACTGCGTGGCCTTCCCGTCCTGCGGCCCCCTT
TCCTCCGCTTGCTGTTTAAATGCTCGTCCTCGTTTTCCTTCCTGCCGCATCCCCGTATATTTTAAGGAGAGGGAAACAGGC
GTGAGTTGGACGGCATGAAAGTTCTCGGCCTGTATGCCGGTTGTGGCGGTCGTTTGCAGATTGCTTTTTTCTTCGAATCGG
TGCTGTAACCCTCGCGAAGAACGACGCTGCAAACGACTTCTCGAACTCTCAGTCGTGTACTTTACGTGCTTCCTTTCAGGG
ACCTCCCCCCGCGTTACTCATTTGTATTCACAGCTACGAAGTGTCTTGCAAGGTGGATTTCTGCCAGGCTCCATGTCTCAC
TCGTTGCGTTTTCGGAAAAGTTCATTGTGAACGTTCCCCTTGCGTGTCATGACTTTATCAG

>UPRT7, 459bp, AY143148.1 *Toxoplasma gondii* strain B7 type II uracil phosphoribosyl transferase (UPRT) gene, intron 7

GTAAGAAATAACGCTTTGAAGGTCGTTGGCTTGTATTACCGTCACTACTGCGACGACATACTGGAGAACTCGAGCATCCTT
TATGGGTTTGTGTTGATGAACGTAGGACTACCAGATATCATTTCTTCATTGGAACACTTTATTGTACACGCATGCCGTTTA
TGCGCTGCTTATCCGAGGGTGGCATCACAGGGACCCTACAAGAAGCTGACGGGGCACCTCCACTATGAAGTGACATGCCAA
TGCATGCCCAGGTTGCATGTTCCAAGTGTCTGGTATCTTTGGCTACGCTGTCTTTTACTAATCGCAAGAAACAGTGCTCGT
CATGGGTGGTAACTCCTTGGTGCCACCAGAACGCATGTGGTGCGTTTGTTGTTCCGCTTTGTTTTCTTGGGAGATTTCGCT
GCGGCCTAATGCAGCCAACCCTTTATGGAATTGTGTGCCCTGCTGTCTTGTCAG

>MIC2-1, 409bp, JX044267.1 *Toxoplasma gondii* strain Me49 micronemal protein (MIC2) gene, intron 1

AAGGTGACGACAACAGTGTACCTCGATTGTGTATCTCGTAGCGTCTCGTCTCCCTCTCTTCGAAGATGGGCAGCAGCAGCC
ATTGTATATCGGAGAGTGTGGTAGTCTCTCTTGAACAGTAAACGGCATTCAGCATGTTTTGTGGTTGCTGGCTTCTGCAGC
TACGTGTTGTTGATGTTGGCTTCTCCCGCACTAGACGTATGGGAAGTTATTTACCGTGGAGCCGCAGTCGGCGTCCTCCCC
CAAGAGACTTGTATTTTCTGTTTTTAACTTTCGTGACTTCACCGGCGAAGACTTCGAGGTGACTACGCACAGAGGGATCAT
GCTGCTGCGTGTAGTGGCCGCATATGCCATGATTATACCCATGTTGTCGCTGTTCTTTGTGGGATATGTGTGTGCTCGTTC
CTGA

>MIC2-3, 371bp, JX044343.1 *Toxoplasma gondii* strain Me49 micronemal protein (MIC2) gene, intron 3

AAGGTTATCAGAAAAGCGGTTGTGTCGCCTTGTACGAATGAGGCGGTATCAGAAGTCGCGTTTGACTGTAGTGAGGGTCGA
ATTGTGTAGGGATGGGGAGACACAGCAAGCCCAGAAGCATGAAGTTGTTCCACGCTACCTTGTCTGCGCGATCGATTTTCT
CAAAGATCATTGGGTGCAACGCGATATGGCAGTCGTTGCGGGCCAAGAAAACGAATTCTACAAACAGGGTGTACAGGGGAA
ACTGCTCGCACGCGATAATTTTTGAAATGGAAGTTACCCTTGGGGCAGTGCTGCTTCATCTTGCGCAGGCATAGGGATGTT
TAGTATCCGACACGATGAGCCTCCGGGTGTCTGTTTATCTGCTCTTC

>BTUB-1 463bp, chrIX, JX045549.1 *Toxoplasma gondii* strain Me49 beta-tubulin (TUB2) gene, intron 1

GTGAGGGAGAACGGGTTCCGGAGTGCGGGTTCCCGGACGCTGTCTTTTGTCCTGTATTCTGCGTGACAGCTTCGCCAGTGT
AAATGGAGCCGTTTTCCCTGTGGAGAATCGCGGAGAATGGAGGACTTTTCCGAGCTCGCAGGTGCCACCCTCCACCTCGAC
GCAACCAAGTGCACGCACATTTGCCGGGTGGTGACCCTAAGACACCGCAGGTCTACCTGCGCGTTTTTTTCAGTCTTTGCA
CACAGTTGCACCGAAAGTCATGTTTTTTTGCGAAGACATCGGTCGTTCTGGTGGGGGAATACTTCACTCCTGTTCGCGCCTA
TGTGCGCAGACAGGTGTCCACCCTCCGCATACGGGCGTCCGGGTGTTCCTACAAAATGCAGGCGCGGCGACATCGGCCAAG
TTCTTCCGTCATTTCAATTTGTTCTGCGATGTGTGTGTGTGTGGTTTCCTCTCTTCAG

>BTUB-2, 141bp, chrIX, AY143124.1 *Toxoplasma gondii* strain B7 type II TUB2 gene, intron 2

GTAAGTTGTACAGAGTGTGCCGCGCAGTCTGGTTGCAGTGTGCGTCTAGAGATGGCACATTTAGTGTGTCTGGGGGTACAG
CTGTGTTCTCGAAGTCTGCGTGGGCATGAATACTGTTTTTTTGCCTCCTTGGTCCTATAG

>EF1 intron 1, 710bp, TGME49_chrX | *Toxoplasma gondii* ME49

CACATGAAGGTACACCAAAAAACCCTTTTATCGAACACGGAAAAAAAGACACTGGCGGACGCCCCGTCAAACCAAAGAGCC
TACCCTTTGAGAAAGCCTCACTTTCACGGGCGTCGTTATGGCCCTACAGGAAAGAAGGGGGAAATAAACCACAGGGAAGAC
TGCGGGAGTAATCATCGATCAACATTCAAGAGGTGGATTTACGGCCTTCGCTAGATGACTGTGAGTGGGGGGTAAGGCATG
GAAATTTCTGATGTTCGCACAACACGCACCAGAGATAAATGGAGGAATGGCCGAGATAAGTGCGTGGTGAAATATTTTTGT
GGTGCCGCCCACCACTCGCGGTGATGGACAACCAGAAGAAACTCCATGGAACAGCCCCCAAAGATGCAGACTCCGACAAAG
ATACGAGTTCACACTGCAGAGTCTGTTTAGGAAGATAGTACGCTGCTACAGTTTAGAGGGGTTCTTCCCAATGCCGTTCTC
TCTAAAGAAGCCAGTCACGGGGAGTTGACCCTATCAAATCATATGAAAGTTTACACACGACCGCCACGACGTGTCTGATGG
CGGGACAATTTCAACTGAAAGTGCACCTGCGCGCTTCCGTCCCATGGAGGCGGGAACGCGAAGCAGGCTGTTTCACCAAGA
AAGAAGACTCTCCTTTCGCCTACTTCCCTGCGCATCTATTTATTTAAGAAAAGGGTGCATTT

>HP intron 2, 846bp, TGME49_chrIV | *Toxoplasma gondii* ME49

TAATCTTTGTTCCCATGCTTGCTGCGTTTTGGATCCATTCGTCAGTTGGAAGAGAGTGTCGCCTTTCTACCTCGAGGGGGA
TAGGCTCAAGCCCGTCCGAACTGTTTTTTATGGTTGCGAACGATTGTACACCATAGCCTTCAACAGTTGTCGCTAGATGAG
CAGAGGGCCGGTGCCTTCGGTCGGAAAACGTGGGTCAACGAGCAGCTTCTAGGTTCAATGAATACGAGATCCGCTGAGTTC
TTGTGTCACTGTTTTCTCACCTCAGTCTCTTGCCTGCTTTGTCCGGTTCCCTTGCCTGTAAGAACTGTTTCTGCCCCAGGT
GCACAATTGCTCATTCCGCTATCGCCTGAAGGGGGTCCAGGGTTGCGTGCTCAGCAGGCGGTTGAGGAAGCTGCGTGGTTC
GTGGCGTGCAGACACGTTGCCCTGCCTTTCTGGATTTTATTGGAGTACACTACGCCTTTCTGGTCGTACCTGCAGAAACAA
ACAGGAAGGGCAGTATGCTTTTCGGCAGCTGTATTCGAGTGCTGCGATCGAGGGAACTGACTGTATTATTCCTGCAGCTTC
GCTGGTTGTGGTGTGCCTCGTTTGCCTTCACTTTTTGCCGAGTCTTTACGGTTCGTGTTATAGTCACTTTTGCGGGCTCAC
GCCACATCATTCTTTCTGTTTCATCAATCTTTCCATCTTTGTTAACTCACGGGGTGCGTGCCTAGCGCGGAAGGGGAGAGT
CAACTGTTTCCTGTTTCCGGTTAGTTTCGTTTGCTCTTCTTGTGCTCATCGTGTTCTGCTCATTTTTTCGCATTTCCGGGT
GTGTAATCGGTGACATATTCTCTGCGTGTTTCTGTC

3.6. REFERENCES

AJZENBERG, D.; COLLINET, F.; MERCIER, A.; VIGNOLES, P.; DARDÉ, M. L. Genotyping of *Toxoplasma gondii* isolates with 15 microsatellite markers in a single multiplex PCR assay. **Journal of Clinical Microbiology**, v. 48, n. 12, p. 4641-4645, 2010.

DUBEY, J. P. **Toxoplasmosis of animals and humans**. 2. ed. Boca Raton: CRC Press, 2010. 336 p.

JIANG, T.; SHWAB, E. K.; MARTIN, R. M.; GERHOLD, R. W.; ROSENTHAL, B. M.; DUBEY, J. P.; SU, C. A partition of *Toxoplasma gondii* genotypes across spatial gradients and among host species, and decreased parasite diversity towards areas of human settlement in North America. **International Journal for Parasitology**, v. 48, n. 8, p. 611-619, 2018.

KHAN, A.; FUX, B.; SU, C.; DUBEY, J. P.; DARDÉ, M. L.; AJIOKA, J. W.; ROSENTHAL, B. M.; SIBLEY, L. D. Recent transcontinental sweep of *Toxoplasma gondii* driven by a single monomorphic chromosome. **Proceedings of the National Academy of Sciences**, v. 104, n. 37, p. 14872-14877, 2007.

LEHMANN, T.; MARCET, P. L.; GRAHAM, D. H.; DAHL, E. R.; DUBEY, J. P. Globalization and the population structure of *Toxoplasma gondii*. **Proceedings of the National Academy of Sciences**, v. 103, n. 30, p. 11423-11428, 2006.

MONTOYA, J. G.; LIESENFELD, O. Toxoplasmosis. **The Lancet**, v. 363, n. 9425, p. 1965-1976, 2004.

PÉREZ-LOSADA, M.; CABEZAS, P.; CASTRO-NALLAR, E.; CRANDALL, K. A. Geographical patterns of *Toxoplasma gondii* genetic diversity revealed by multilocus PCR-RFLP genotyping. **Infection, Genetics and Evolution**, v. 16, p. 38-53, 2013.

SHWAB, E. K.; ZHU, X. Q.; MAJUMDAR, D.; PENA, H. F. J.; GENNARI, S. M.; DUBEY, J. P.; SU, C. Geographical patterns of *Toxoplasma gondii* genetic diversity revealed by multilocus PCR-RFLP genotyping. **Parasitology**, v. 141, n. 4, p. 453-461, 2014.

SU, C.; SHWAB, E. K.; ZHOU, P.; ZHU, X. Q.; DUBEY, J. P. Moving towards an integrated approach to molecular detection and identification of *Toxoplasma gondii*. **Parasitology**, v. 137, n. 1, p. 1-11, 2010.

SU, C.; KHAN, A.; ZHOU, P.; MAJUMDAR, D.; AJZENBERG, D.; DARDÉ, M.-L.; ZHU, X. Q.; AJIOKA, J. W.; ROSENTHAL, B. M.; DUBEY, J. P.; SIBLEY, L. D. Globally diverse *Toxoplasma gondii* isolates comprise six major clades originating from a small number of distinct ancestral lineages. **Proceedings of the National Academy of Sciences**, v. 109, n. 15, p. 5844-5849, 2012.

VINCZE, T.; POSFAI, J.; ROBERTS, R. J. NEBcutter: A program to cleave DNA with restriction enzymes. **Nucleic Acids Research**, v. 31, n. 13, p. 3688-3691, 2003.

WALKER, B. J.; ABEEL, T.; SHEA, T.; PRIEST, M.; ABOUELLIEL, A.; SAKTHIKUMAR, S.; CUOMO, C. A.; ZENG, Q.; WORTMAN, J.; YOUNG, S. K.; EARL, A. M. Pilon: An integrated tool for comprehensive microbial variant detection and genome assembly improvement. **PLoS ONE**, v. 9, n. 11, p. 1-14, 2014.

REFERENCES

BATZ, M. B.; HOFFMANN, S.; MORRIS JR., G. **Ranking the risks**: The 10 pathogen-food combinations with the greatest burden on public health. Gainesville: Emerging Pathogens Institute, University of Florida, p. 70.

BOSSI, P.; BRICAIRE, F. Severe acute disseminated toxoplasmosis. **The Lancet**, v. 364, n. 9434, p. 579, 2004.

DUBEY, J. P. Isolation of *Toxoplasma gondii* from the feces of a helminth free cat. **The Journal of Protozoology**, v. 15, n. 4, p. 773-775, 1968.

DUBEY, J. P. **Toxoplasmosis of animals and humans**. 2. ed. Boca Raton: CRC Press, 2010. 336 p.

DUBEY, J. P.; LAGO, E. G.; GENNARI, S. M.; SU, C.; JONES, J. L. Toxoplasmosis in humans and animals in Brazil: High prevalence, high burden of disease, and epidemiology. **Parasitology**, v. 139, n. 11, p. 1375-1424, 2012.

HILL, D. E.; DUBEY, J. P. *Toxoplasma gondii*: Transmission, diagnosis and prevention. **Clinical Microbiology and Infection**, v. 8, n. 10, p. 634-640, 2002.

HOWE, D. K.; SIBLEY, L. D. *Toxoplasma gondii* comprises three clonal lineages: Correlation of parasite genotype with human disease. **The Journal of Infectious Diseases**, v. 172, n. 6, p. 1561-1566, 1995.

LEHMANN, T.; MARCET, P. L.; GRAHAM, D. H.; DAHL, E. R.; DUBEY, J. P. Globalization and the population structure of *Toxoplasma gondii*. **Proceedings of the National Academy of Sciences**, v. 103, n. 30, p. 11423-11428, 2006.

MONTOYA, J. G.; LIESENFELD, O. Toxoplasmosis. **The Lancet**, v. 363, n. 9425, p. 1965-1976, 2004.

MOURA, L.; BAHIA-OLIVEIRA, L. M. G.; WADA, M. Y.; JONES, J. L.; TUBOI, S. H.; CARMO, E. H.; RAMALHO, W. M.; CAMARGO, N. J.; TREVISAN, R.; GRAÇA, R. M. T.; SILVA, A. J.; MOURA, I.; DUBEY, J. P.; GARRETT, D. O. Waterborne toxoplasmosis, Brazil, from field to gene. **Emerging Infectious Disease**, v. 12, n. 2, p. 326-329, 2006.

PAPPAS, G.; ROUSSOS, N.; FALAGAS, M. E. Toxoplasmosis snapshots: Global status of *Toxoplasma gondii* seroprevalence and implications for pregnancy and congenital toxoplasmosis. **International Journal for Parasitology**, v. 39, n. 12, p. 1385-1394, 2009.

PENA, H. F. J.; GENNARI, S. M.; DUBEY, J. P.; SU, C. Population structure and mouse-virulence of *Toxoplasma gondii* in Brazil. **International Journal for Parasitology**, v. 38, n. 05, p. 561-569, 2008.

SHWAB, E. K.; ZHU, X. Q.; MAJUMDAR, D.; PENA, H. F. J.; GENNARI, S. M.; DUBEY, J. P.; SU, C. Geographical patterns of *Toxoplasma gondii* genetic diversity revealed by multilocus PCR-RFLP genotyping. **Parasitology**, v. 141, n. 4, p. 453-461, 2014.

SU, C.; ZHANG, X.; DUBEY, J. P. Genotyping of *Toxoplasma gondii* by multilocus PCR-RFLP markers: A high resolution and simple method for identification of parasites. **International Journal for Parasitology**, v. 36, n. 7, p. 841-848, 2006.