Comparative Analysis of Existing Pipelines for Assessment of Arbuscular Mycorrhizal Fungal Biodiversity in Natural and Commercial Rooibos (*Aspalathus linearis*) and Honeybush (*Cyclopia intermedia*) Soil Samples.

*****

A mini-thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE AT RHODES UNIVERSITY

by
Coursework / Thesis
in
Bioinformatics and Computational Molecular Biology
In the Department Of Biochemistry, Microbiology & Biotechnology
Faculty of Science

by Hermina Johanna de Wit

February 2015

*****

Abstract

There is a mutually beneficial association between the fungal phylum Glomeromycota and higher plant roots. Arbuscular mycorrhizal (AM) fungi are important in performing various ecological functions in exchange for host photosynthetic carbon in order to contribute to the fitness of the host plant. Conclusions of many studies stated that AM fungi are essential for the fitness and study of many higher plants in their natural environments in order to achieve a holistic understanding of different ecosystems in which these fungi are present (Willis *et al*., 2012).

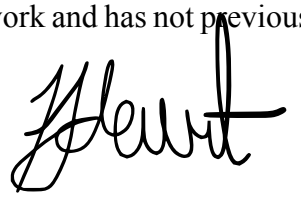
The two types of plant hosts looked at were Rooibos and Honeybush. Rooibos (*Aspalathus linearis*) and Honeybush (*Cyclopia intermedia*) are both popular teas in South Africa and both have a growing worldwide market because of their medicinal properties since they are rich in polyphenols. Primers were designed specifically for 454 FLX pyrosequencing of the samples. Samples were taken from both natural and commercial environments for both host plant types in order to determine the differences in the AM fungal diversity and abundance within the different samples. The 18S ribosomal sequences for both the natural and commercial samples of Rooibos and Honeybush of AM fungi needed to be rationalized and a pipeline needed to be identified or created in order to study these types of fungi by classifying and sorting the sequencing data.

The expected outcome was to find that there were a higher diversity and abundance of fungi in the natural than in the commercial samples. This could have been a result of the unnatural growth environments of the Rooibos and Honeybush plantations, and also due to the use of pesticides and not purposefully trying to maintain the AM fungi cultures within these environments. The outcome of this study lead to the realization that there are still not enough sequencing data of AM fungi available in the present databases. This lead to difficulties designing a pipeline and analyzing the sequencing data produced by Rhodes University. Further studies will need to be done in order to find a suitable pipeline, or even design a suitable pipeline, to be able to effectively analyse 18S AM fungal data.

Declaration

The research described in this thesis was carried out as part of the one-year MSc coursework and research thesis programme in Bioinformatics and Computational Molecular Biology, from 15 July 2015 to 31 January 2016 under the supervision of Prof Özlem Taştan Bishop and Prof Joanna Dames.

I, Hermina Johanna de Wit, declare that this thesis submitted to Rhodes University is wholly my own work and has not previously been submitted for a degree at this or any other institution.

Signature: 

Date: February 2016

Acknowledgements

I would like to express my gratitude to both my supervisors, Prof Özlem Taştan Bishop and Prof Joanna Dames, for their guidance when doing research and writing my MSc thesis. I would also like to thank the CBIO unit at the University of Cape Town, and especially Gerrit Botha and Dr Katie Viljoen for their knowledge in using the QIIME pipeline, analyzing metagenomic data and willingness to help when I had questions; and Prof Oleg Reva from the University of Pretoria for his knowledge in various programs to analyse metagenomic data.

I would also like to thank my colleagues at RUBi for their support, help and smiles.

I would like to thank Dr Gwynneth Matcher from the Department of Biochemistry and Microbiology at Rhodes University who was always available to answer questions I had regarding the raw sequencing files and other sequencing files I needed.

I would like to thank the RDP Staff (rdpstaff@msu.edu) for helping with questions I had regarding the RDP pipeline and their quick responses at any time of day. I would like to thank Benli Chai of the RDP Staff from the Center for Microbial Ecology at Michigan State University for his quick response, help and knowledge regarding the RDP pipeline.

I would like to thank my mother, Ermie, for her constant support and never giving me a chance to give up when I felt I have failed at something. Thank you for all the calls, motivation, laughter and tears. Thank you for teaching me to never give up during all my years of study and all the times of hardships. I would also like to thank my uncle, JC, for always standing by me and being the best second dad I could ask for. Also, thank you to my darling grandmother, Hermien, who stood by us through all the trying times.

I would like to thank all my friends in Johannesburg for the extreme support and love I received from you during this past two years and my year away from you. You are my second family.

I would like to thank Rhodes University and the Ada and Bertie Levenstein scholarship and NRF for making it possible for me to make this year a reality. I would like to thank H3ABionet and CBIO for making it possible for me to join your research group for August 2015 and learn from you.

Finally, I would like to thank God for giving me this opportunity to further my studies and to give it my all in a new and unknown field.

Table of Contents

Abstract ..... - 2 -
Declaration ..... - 3 -
Acknowledgements ..... - 4 -
Table of Contents ..... - 5 -
List of Abbreviations ..... - 7 -
List of Webservers and Applications ..... - 8 -
Chapter 1: Literature Review ..... - 10 -
1.1) Introduction ..... - 10 -
1.2) Methods used to identify fungal species ..... - 11 -
1.3) Taxonomic diversity of AM fungi ..... - 12 -
1.4) Genetic diversity of AM fungi ..... - 15 -
1.5) Functional and ecological role of AM fungi ..... - 15 -
1.6) Background of Rooibos and Honeybush ..... - 17 -
1.7) Why can metagenomics be used in this study? ..... - 18 -
1.8) Comparisons of different tools in other metagenomic pipelines, as well as individual tools that can be used in future studies ..... - 20 -
1.9) Problem statement ..... - 26 -
1.10) Aims ..... - 26 -
1.11) Objectives ..... - 27 -
Chapter 2: Initial Processing of Reads ..... - 28 -
2.1) Overview ..... - 28 -
2.2) Introduction ..... - 29 -
2.2.1) QIIME ..... - 29 -
2.2.2) Ribosomal Database Project (RDP) ..... - 30 -
2.2.3) Mothur ..... - 31 -
2.3) Methodology ..... - 32 -
2.3.1) QIIME ..... - 32 -
2.3.2) RDP pipeline ..... - 38 -
2.3.3) Mothur ..... - 40 -
2.3.4) Comparisons of the initial processing results ..... - 42 -
2.3.5) NextGen Workbench analysis ..... - 42 -
2.4) Results and Discussion ..... - 42 -
2.4.1) QIIME ..... - 43 -
2.4.2) RDP pipeline ..... - 67 -
2.4.3) Mothur ..... - 76 -
2.4.4) Comparisons of QC results from different pipelines ..... - 79 -
2.4.5) NextGen Workbench ..... - 90 -
Chapter 3: Classification and further processing of the sample reads ..... - 90 -
3.1) Overview ..... - 90 -
3.2) Introduction ..... - 91 -
3.2.1) OTU picking, classification and phylogenetic tree generation ..... - 91 -
3.2.2) Measure diversity and other statistical tests ..... - 92 -

3.3) Methodology ..... - 93 -
3.3.1) QIIME ..... - 93 -
3.3.2) RDP pipeline ..... - 96 -
3.3.3) Mothur ..... - 97 -
3.3.4) Generating phylogenetic trees with Jalview and MEGA7 ..... - 101 -
3.3.5) Optional methods ..... - 102 -
3.4) Results and Discussion ..... - 102 -
3.4.1) QIIME ..... - 102 -
3.4.2) RDP pipeline ..... - 105 -
3.4.3) Mothur ..... - 109 -
3.4.4) Generating phylogenetic trees ..... - 110 -
3.4.5) Optional methods ..... - 128 -
Conclusion ..... - 129 -
References ..... - 133 -
Appendices ..... - 142 -
Appendix I: QIIME code ..... - 142 -
Appendix II: Mothur code ..... - 146 -
Appendix III: Optional method nr 1 ..... - 149 -
Appendix IV: Optional method nr 2 ..... - 161 -
Appendix V: Optional method nr 3 ..... - 164 -
Appendix VI: The source code for Cutadapt ..... - 172 -
Appendix VII: Optional method nr 4 ..... - 181 -

List of Abbreviations

AM – Arbuscular Mycorrhizae

bp – Base Pairs

GCPRS – Genealogical Concordance Phylogenetic Species Recognition

HM – Heavy Metal

ITS – Internal Transcribed Spacer

LSU – Large Subunit

MAFFT - Multiple Alignment using Fast Fourier Transform

MSA – Multiple Sequence Alignment

NGS – Next Generation Sequencing

OS – Operating System

OTU – Operational Taxonomic Unit

PCR – Polymerase Chain Reaction

QC – Quality Control

QIIME – Quantitative Insights Into Microbial Ecology

RDP – Ribosomal Database Project

RUBi – Rhodes University Bioinformatics Unit

SSU – Small Subunit

VM – Virtual Machine (another name for Virtualbox)

VT – Virtual Taxa

List of Webservers and Applications

QIIME

http://qiime.org

RDP pipeline

https://rdp.cme.msu.edu

Mothur

http://www.mothur.org

FunGene – Functional gene pipeline and repository

http://fungene.cme.msu.edu

Unipro UGENE

http://ugene.net

MaarjAM 18S database

http://maarjam.botany.ut.ee

Python

https://www.python.org

UNIX

http://www.unix.org

Virtualbox

https://www.virtualbox.org

Jalview

http://www.jalview.org

MEGA7 (Mac OS)

http://www.megasoftware.net/webhelp/walk_through_mega/mega_basics_hc.htm

MEGAN5

http://ab.inf.uni-tuebingen.de/software/megan5/

MUSCLE

http://www.drive5.com/muscle/

http://www.ebi.ac.uk/Tools/msa/muscle/

MAFFT

http://mafft.cbrc.jp/alignment/software/

http://www.ebi.ac.uk/Tools/msa/mafft/

GenBank

http://www.ncbi.nlm.nih.gov/genbank/

CLC Genomics Workbench 8

http://www.clcbio.com/products/clc-genomics-workbench/

NextGen Workbench

http://www.dnabaser.com/download/nextgen-fastq-editor/

R and RStudio

https://www.r-project.org

https://www.rstudio.com

BioPython

http://biopython.org

Chapter 1: Literature Review

1.1) Introduction

The fungal kingdom is a very diverse kingdom with species ranging from unicellular yeasts, that are microscopic, to large mushrooms. There are a great number of fungi that have been described and it is estimated by doing high-throughput sequencing that there are roughly 5.1 million different species (Blackwell, 2011). Spores are the main method of dispersal that differ in size, morphology and the method they are released from the fungal hyphae. Fungi are very important in the ecosystem because they have positive influences in the drug and food industry, for instance; and negatively, because of their role as plant and animal pathogens (Tonge *et al*., 2014). Figure 1.1 shows a micrograph of Arbuscular Mycorrhizal (AM) fungi within the host plant root with its respective arbuscules, vesicles and intercellular hyphae.

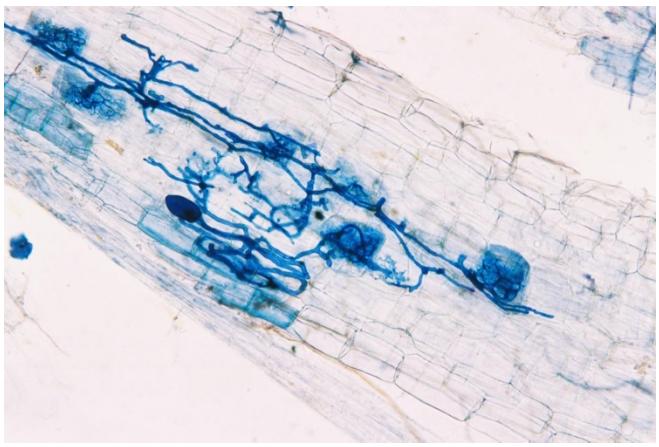


Figure 1.1: A micrograph of AM fungi within the host plant root. Arbuscules (blue aggregates) can be seen within the root cells. These arbuscules are important for the transfer and storage of nutrients (Department of Microbiology and Biochemistry, Rhodes University).

AM fungi were specifically looked at during this study. AM fungi are endosymbionts which are found within the roots of most vascular and terrestrial plants (Wang and Qui, 2006). AM fungi contribute to the host plant's health by enhancing its nutrient uptake. This is especially helpful in areas where the nutrients in the soil are depleted (Bolan, 1991; Clark and Zeto, 2000). AM fungi reduce the effects of soil-borne plant pathogens, reduce the host's uptake of heavy metals, and improves the host plant's water balance. AM fungi contribute all these functions to the host plant in exchange for host plant carbon (Willis *et al*., 2012). A schematic representation of this exchange can be seen in section 1.5 where it is explained in more detail. AM fungi also assist in concentrating carbon in the soil, thus contributing towards the balance of carbon sinks (Wright and Upadhyaya, 1998).

When referring to the taxonomy of AM fungi, there are surprisingly few recognized species, considering that the association has been present for the past 450 million years. There are only 240 species in 4 orders, 9 families and 18 genera in the class Glomeromycetes in the phylum Glomeromycota (Muthukumar *et al*., 2009; Schüßler and Walker, 2012) that are recognised.

Spores are not the only infective unit in soils, other propagates are the mycorrhizal colonizing root fragments and hyphae networks (Smith and Read, 2008). Glomeromycota spores are multi-nucleated, heterokaryotic and formed asexually. This gave rise to questions regarding speciation and adaptation (Redecker and Raab, 2006).

1.2) Methods used to identify fungal species

Culture-based methods are mainly used to identify microscopic fungi, but these methods greatly underestimate the diversity of the fungi. Hawksworth (1991) estimated that only 17% of fungi can be readily grown in culture. However AM fungi cannot be grown in culture, AM spores can be extracted from soil. This technique microscopically distinguishes differences but is limited, resulting in the underrepresentation of the populations (Tonge *et al*., 2014). Techniques using PCR have helped to develop means for identifying fungi without the need of culture-based methods. Amplicon-based techniques that target various regions of the nuclear ribosomal operon (rDNA) have been applied because this rDNA region contains multiple copies, highly conserved and variable regions, and numerous universal primers do exist in order to amplify these regions (Tonge *et al*., 2014). In this case, the 18S gene of AM fungi were amplified by the Department of Microbiology and Biochemistry at Rhodes University in 2014.

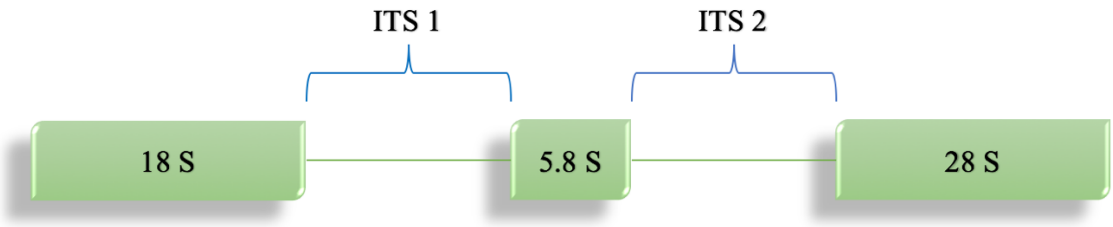


Figure 1.2: Representation of the fungal ribosomal operon structure that consists of various parts. The small subunit SSU (18S) and LSU (28S) are separated by ITS regions (ITS 1 and ITS 2) that surround the 5.8S gene (Lindahl *et al*., 2013).

In early culture-free studies of fungi, conserved oligonucleotide primers were used to amplify fungal DNA. Sanger sequencing was used to sequence the amplicons (Sanger *et al*., 1977) and provided long enough sequences of length ~ 1000 bp with base-calling accuracy that was

sufficient for such analysis, but the sequencing data was not adequate enough to resolve complex microbial systems. The introduction of next generation sequencing techniques increased the amount of sequencing data and thus the sequencing depth but this was at the expense of much shorter reads being generated (Tonge *et al*., 2014). The recent availability of bench-top sequencers with increased amplicon lengths, has placed the ability to generate thousands of longer reads in the hands of small and medium sized research laboratories (Chan *et al*., 2012).

There are three common DNA markers that are currently being used: the small subunit (SSU) rRNA gene, the internal transcribed spacer (ITS) gene, and the large subunit (LSU) rRNA gene. The SSU is mostly used during ecological studies (Helgason *et al*., 1999) whereas the ITS and LSU regions are mostly used in taxonomic identifications and constructions of the Glomeromycota (Redecker *et al*., 2013). SSU rDNA is mostly used for phylogenetic analysis of AM fungi because of the ITS region. The ITS region is mostly used for identifying and barcoding the fungi (Lloyd-Macgilp *et al*., 1996); but this method was also found to underestimate the diversity of Glomeromycota (Husband *et al*. 2002).

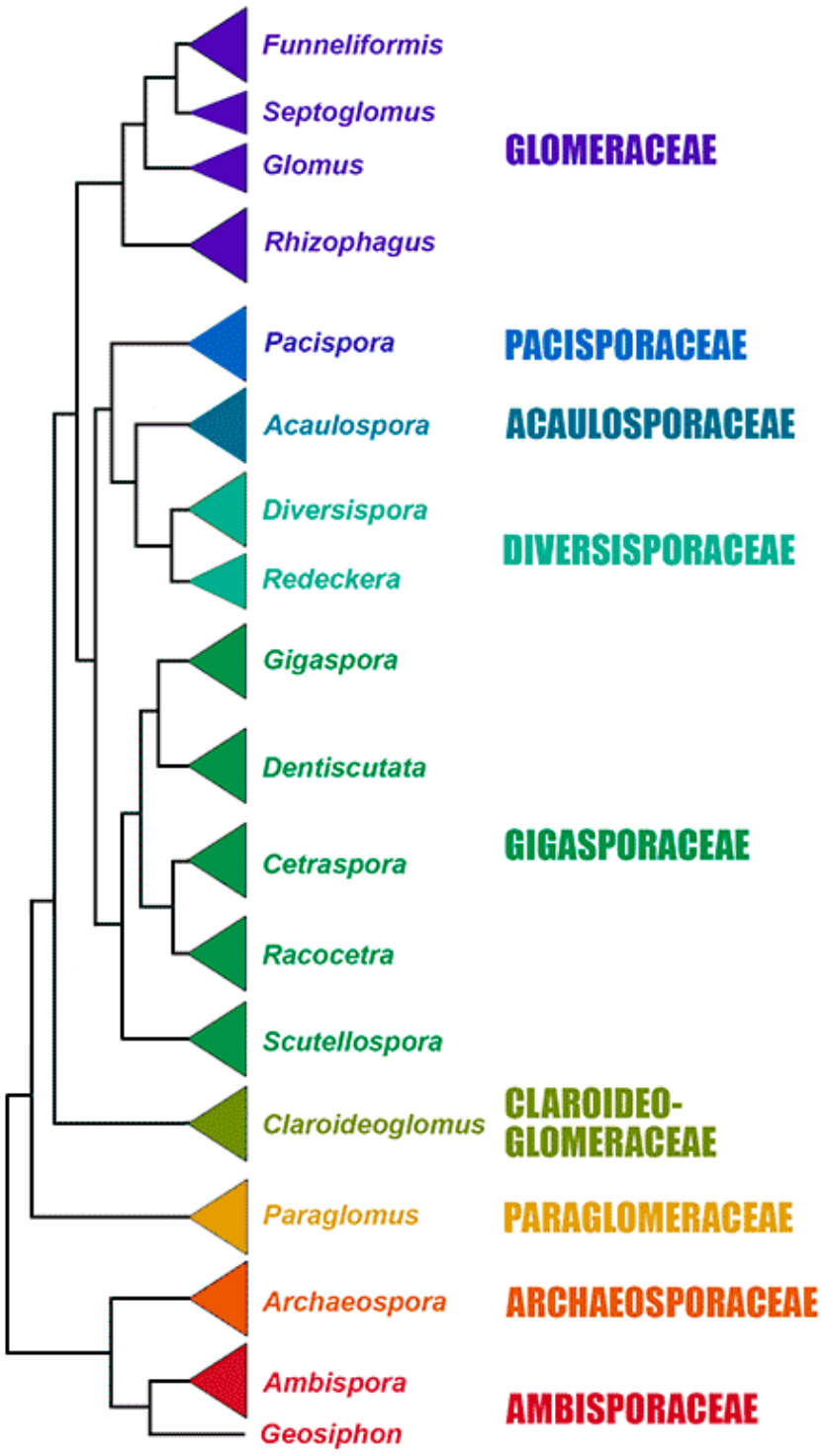
1.3) Taxonomic diversity of AM fungi

Fossil records and molecular data have shown that AM fungi have been present for millions of years, even in ancient terrestrial plants (Krüger *et al.,* 2012). The species diversity of fungi is very low with only 240 recognized species (Schüßler and Walker, 2012). Molecular studies have shown that there is more diversity in AM fungi than was initially suspected (Redecker, 2002). Law (1985) suggested that the low species diversity was probably due to the mutualistic symbiotic relationship between the fungi and the host plant as well as the fungi having no need for speciation due to very little or no selective pressures. Taxonomic classification of AM fungi have been based on the identification and classification of its spores size, shape, colour and the presence of various cell wall layers (Morton, 1988; Morton and Benny, 1990). The main problem with collecting from soil, is the spores are generally degraded and not all species produce spores at the same time. Some AM species also do not produce spores at all and this limits our ability to study population diversity (Law, 1985).

AM fungi are also known as Glomeromycota (taxonomy shown in Figure 1.3 below). Glomeromycota spores may have different morphology and functional structures. For instance, in *Glomus intraradices*, the spore wall is simple, whereas in *Scutellaspora* spp. spores, there

are several membranes and they have complex germination shields. The discovery of the dimorphic species in *Glomus* and *Acaulospora*, showed that they can be homologous (Morton and Redecker, 2001).

AM fungi were originally placed in six genera of the order Glomales (phylum Zygomycota) using various phylogenetic analysis (Morton and Benny, 1990). Zygomycota were interpreted as azygospores, which have one gametangium. Later it was noted that AM fungi produce a merosporangium with one spore. Using rRNA analysis, it was found that most AM fungi are within specific monophyletic clades separated from other major fungal groups. The fungal groups were so diverse that they were placed into a new phylum called Glomeromycota (Schüßler *et al.*, 2001). Krüger *et al*. (2012) stated that after they studied 136 species of AM fungi and 27 undescribed species, they found that the undescribed species of AM fungi were underestimated. SSU rDNA is mostly used for phylogenetic analysis of AM fungi because of the ITS region. The ITS region is mostly used for identifying and barcoding the fungi (Lloyd-Macgilp *et al*., 1996); but this method was also found to underestimate the diversity of Glomeromycota (Husband *et al*. 2002).



http://schuessler.userweb.mwn.de/amphylo/Schuessler&Walker2010_Glomeromycota.pdf

Figure 1.3: Illustration of the phylogeny of Glomeromycota. This was based on the SSU RNA sequences and showed the relationships among the orders and genera. Families were indicated by distinct colours. (Krüger *et al*., 2012, Oehl *et al*., 2008; Schüßler and Walker, 2010).

Ecological studies of AM fungi have been reliant on spore samples. Identification of AM fungi in soil samples became important seeing that the data from the soil samples were only correct

for that specific sample. Distinguishing between the active symbionts within a plant and the spore communities in the soil samples was a very important challenge. The difficulties in identifications have lead to the development of specific primers for all lineages for different PCR techniques which lead to more accurate system analysis (Wubet *et al*., 2006; Rosendahl *et al*., 2009).

1.4) Genetic diversity of AM fungi

Molecular analysis and studies of plant fossils indicate that AM fungi were present since the Ordovician period where the first land plants had a symbiotic relationship with these fungi. It was noted in the paper by Rosendahl (2008) that these fungi might have been essential for the adaptation of plants to the terrestrial environment. By using various PCR techniques with specific primers, it was found that the diversity of AM fungi in soil samples were much more diverse than originally expected (Gallotte *et al*., 2004). DNA polymorphisms have been used to identify the geographic origin of different AM fungi (Sanders *et al*., 1995). AM fungi were regarded as multigenomic seeing that thousands of nuclei exist within one spore (Kuhn *et al.*, 2001). Heijri and Sanders (2005) found that the genetic diversity and polymorphisms were inherited only in one nuclei and not shared between different nuclei within a spore.

The genetic variations within the AM species are not yet well understood but it can be inferred that the functional diversity of the host plant was a result of the genetic variations. Recombination within the AM spores have not been proven yet but most of the evidence suggested that fungal evolution has occurred asexually. This lack of certainty negates the use of the GCPRS (Genealogical Concordance Phylogenetic Species Recognition) to identify Glomeromycota species (Rosendahl, 2008).

1.5) Functional and ecological role of AM fungi

Plant-soil interactions cannot be studied without considering the interactions between the plant and fungi (Rosendahl, 2008). The functions of AM symbiosis can differ significantly between host plants and different isolates (Smith and Read, 2008). These fungi are present from the sub-polar regions to the tropical rain forests and some aquatic ecosystems (Rosendahl, 2008). Fungi mostly have an effect on plant growth. The most important being the exchange between the host plant roots and AM fungus of phosphorus and the photosynthate (van der Heijden *et al*., 1998). The symbiotic efficiency of the host plant roots and the AM species can be defined as "the amount of carbon gained by a growth response from the symbiotic relationship with AM fungi minus the amount of carbon lost by investment in maintaining the symbiotic relationship"

(Lee *et al*., 2013). In most studies, the presence of the fungal symbionts lead to increased plant growth, but there were some instances where the plant lost biomass due to AM colonization (Constantini *et al*., 2014), for example under shading and reduced photosynthesis. Angelard *et al*. (2010) found that the inoculation of rice plants with AM fungi increased the hosts biomass by up to 5 times. Van der Heijden *et al*. (1998) also stated that AM fungi influenced the composition and development of plant communities.

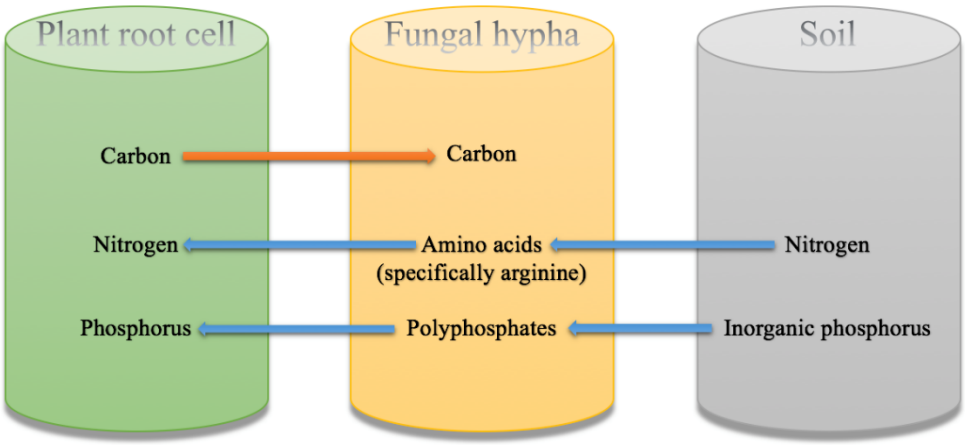


Figure 1.5: Representation of the flow of nutrients between the soil, fungi and plant. AM fungi produce structures called arbuscules within root cells to exchange nutrients. This figure showed roots of *Allium porrum* colonized by *Glomus mosseae* (this was an example of AM fungal host colonization). This representation shows that there is a symbiotic relationship between the host and the fungi where there is a reciprocal nutrient exchange. The host also receives protection against abiotic and biotic stresses in the soil environment. (Selosse and Rousset, 2011).

As shown in figure 1.5, extraradical mycelia absorb mineral compounds from the soil. It was found that there were differences in the uptake and supply of soil phosphorus and nitrogen to plants due to different concentrations of P-transporters and N-assimilation genes in the AM species (Maldonado-Mendoza *et al*., 2001). Genre *et al*. (2008) found that the very specific signal exchange between the fungi and the host plant roots are important in the completion of the AM fungal life cycle. Ravnskov and Jakobsen (1995) stated that many of the fungi show low host specificity and that different species may differ in their effects on plant growth. This can be due to differences in the species and their ability to acquire nutrients, which in turn provide the host plant with stress tolerance, as well as tolerance towards draught and pathogens (Rosendahl, 2008).

Fungi have colonised a wide variety of ecosystems and terrestrial plant roots. About 20% of the carbon compounds of the host plants are exchanged for inorganic minerals by AM fungi.

AM fungi play important roles in the circulation of phosphorus and nitrogen, saline and heavy metal tolerance, increased protection against nematodes and various root diseases of the host plant rhizosphere (Linderman, 2000). The root systems that have been colonised by AM fungi have a more dense hyphal network and this, in turn, increases the plant's biomass (Lee *et al*., 2013).

Understanding the functioning of AM fungi can help prevent possible stresses due to heavy metal (HM) pollution in contaminated soils close to industrial areas by using the principle of phytoremediation. There is evidence that some Glomeromycota species have beneficial influences, under certain conditions, on plants growing in soils contaminated with HMs. Fungi produce glomalin which binds to the HMs copper, cadmium and lead. It was reported by Joner *et al*. (2000) that HMs can bind to chitin when the hyphal walls are in close proximity. AM fungi are also reported to produce chelators that bind to the metals which are pumped out of the cells by HM transporters. A phenomenon of spore tolerance also occurs which can lead to the adaptation of the fungi when they are present in different environments (Willis *et al*., 2012).

1.6) Background of Rooibos and Honeybush

The two types of plant hosts previously researched and sequenced via 454 FLX pyrosequencing by the Department of Biochemistry and Microbiology at Rhodes University were Rooibos (*Aspalathus linearis*) and Honeybush (*Cyclopia intermedia*). Rooibos and Honeybush are natively grown in South Africa and are very popular native teas. These teas have a growing worldwide market. Both of these teas have been used in traditional medicine because they are rich in polyphenols. Rooibos is a rare source of dihydrochalcones, aspalathin and nothofagin which are important dietary supplements. Honeybush includes the polyphenols xanthone and mangiferin as well as the flavonones hesperitin and isokuranetin. Both Rooibos and Honeybush share potent antioxidants and contribute to antimutagenic activities *in vitro*. Rooibos and Honeybush have been tested on animals and the results showed that both these teas possess potent antioxidants, immune-modulating and chemopreventative attributes. Human studies, on the other hand, are very limited for Rooibos and no human studies have been conducted for Honeybush. As yet, no adverse effects of these two teas have been reported (McKay and Blumberg, 2007).

Rooibos (also called *A. contaminata*, *A. corymbosus*, *Borbonia pinifolia* and *Psoralea linaeris*) is indigenous to the Cedarberg in the Western Cape province of South Africa. It is extensively grown in this area for its use as a herbal tea. It is a leguminous shrub. Post-harvest the leaves

and stems are fermented before drying, or dried immediately after harvesting. Sometimes the leaves are left unfermented and this product is then called green Rooibos. During the fermentation process, the colour of the leaves and stems changes from green to red due to the oxidation of polyphenols. The final product is red tea leaves. Other names for this tea include Rooibos tea, Rooibosch, Rooitea or Rooitee (McKay and Blumberg, 2007).

Honeybush is also a leguminous woody shrub that grows on the mountain slopes of Langkloof between the Eastern and Western Cape regions in South Africa. This tea is not widely cultivated and the commercially available products are collected from natural plant populations. The Honeybush leaves, stems and flowers are used to make this herbal tea. The plant material is cured in a heap or at elevated temperatures in an oven and then allowed to dry. Fermentation causes the plant material to change colour from green to dark brown during the oxidation of the phenolic compounds. Honeybush tea can be referred to as Heuningtee, Bergtee, Boertee, Bossietee and Bush tea (McKay and Blumberg, 2007).

1.7) Why can metagenomics be used in this study?

This project was based on many metagenomics techniques, programs and pipelines and so it was fit to consider what metagenomics is: metagenomics is a study that applies genomic techniques to study genetic diversity of life on Earth. But in this instance, the need for isolation and laboratory cultivation of individual organism samples were not done because AM fungi cannot be grown in culture. In order to study the AM population, DNA extraction from the AM fungi in the soil samples were done and amplified using 454 FLX pyrosequencing by the Department of Microbiology and Biochemistry at Rhodes University in 2014. Metagenomics were used to provide unprecedented access to the AM fungal biodiversity of the Rooibos and Honeybush samples. The amplicon-based approach had to be ubiquitous in the taxonomic range of interest as well as variable enough in order to discriminate between different species (Santamaria, 2012). The 18S amplicon was selected.

In order to sequence reads for Rooibos and Honeybush (both natural and commercial samples), 454 FLX pyrosequencing was used. This technique is based on the use of pyrosequencing where it uses chemiluminescence to detect nucleotides. The light signal is detected when a complimentary base is incorporated into the sequencing strand and light is generated by luciferase. 454 pyrosequencing (Yadav *et al*., 2015) is a widespread technique in that it can be used with genome re-sequencing, variant detection, *de novo* genome sequencing, rRNA amplicon sequencing, ChIPSeq and RNASeq.

The advantages of using pyrosequencing are that reads are mostly above 500 bp with 99% accuracy with a high throughput of 400 – 500 bp per run, it does not rely on cloning efficiency within cultures, and DNA libraries can be barcoded and separated during data analysis. On the other hand, the disadvantages include difficulties with homopolymers and long repeats, and each run is very expensive and not ideal for re-sequencing compared to Illumina and SOLiD sequencing techniques (Fakruddin and Chowdhury, 2012).

Next generation sequencing refers to multiple different types of sequencing, including genome sequencing, genome resequencing, transcriptome profiling (RNA-sequencing), and DNA-protein interactions (ChIP-sequencing) (de Magalhães *et al*., 2010). Resequencing is usually necessary because the genome of one individual will not indicate the variations that occur within the other individuals of the same species. In recent years, the demand for low-cost sequencing drove the development of next generation sequencing technologies. But these technologies parallelized the sequencing process in order to produce thousands of reads simultaneously (Hall, 2007; Church, 2006).

454 Life Sciences developed a parallelized version of pyrosequencing. This technology was acquired by Roche Diagnostics. 454 pyrosequencing amplifies the DNA inside water droplets in an oil solution. Each of these water droplets contain a single DNA template that is attached to a single primer-coated bead. This then forms a clonal colony. The pyrosequencer contains multiple picoliter-volume wells. Each of these wells contains a single bead and sequencing enzymes. Pyrosequencing used luciferase to generate light in order to detect the different nucleotides being incorporated within the DNA. The combined data is then used to generate the sequence results (Margulies *et al*., 2005).

In a study done by Chengwei *et al*. (2012), they compared the Roche 454 and Illumina platforms for metagenomic studies. They sequenced the same community DNA sample with each platform. These two platforms agreed on over 90% of the assembled contigs and 89% of the unassembled reads. The two platforms also agreed on the estimated gene and genome abundances within the sample. Their findings suggested that both Roche 454 and Illumina were reliable for quantitative assessment of metagenomic samples. The differences were the read lengths, sequencing cost, errors in the consensus sequences, resolving sequences with repetitive structures or palindromes, and which is propably more suitable for metagenomic analyses. Illumina yielded more accurate and longer contigs but the read lengths were shorter that that

produced by Roche 454. The Illumina sequencing data had cost about one fourth of the cost of the Roche 454 data. They stated that Illumina and other short-read sequencing methods, may be more appropriate to do metagenomic studies with. Roche 454 had 14% less complete genes than Illumina and this lead to the higher sequencing error rate associated with A- and T-rich homopolymers. This result also agreed with that found by Margulies *et al*. (2005) and Quince *et al*. (2009). These errors were not observed in the Illumina data possibly due to the high sequence coverage and the less pronounced sequencing biases. Roche 454 sequencing may be better for resolving sequences with repetitive structures or palindromes or metagenomic analyses that are based on unassembled reads. This will be due to the longer read lengths produced in Roche 454 sequencing.

The amplicon-based approach was used to determine the abundance and identity of AM fungi in the four samples (Honeybush natural, Honeybush commercial, Rooibos natural, Rooibos commercial). Amplicon sequencing was used to generate information regarding the taxonomic composition and phylogenetic structure which is used to monitor environmental communities. This technique does have some biases regarding PCR errors, also gene copy number bias, read replicates and chimeric reads. The genes this approach could be applied on were strictly limited to the applied primers. There were also some other known biases when using the amplicon approach (Arigon *et al*., 2007) including mosaicism, intragenomic heterogeneity, lack of a threshold identity, biased amplification, artificial replicates, and chimeric sequences.

Recent NGS (Next Generation Sequencing) advances negated the need for using cloning libraries. Metagenomics is a very useful field of study because it gives the researcher access to many more organisms seeing that most microbes and fungi are unable to be grown under axenic culture conditions - which was also the main problem with AM fungi. Comparative metagenomics (Yadav *et al*., 2015) can also be done in future in order to achieve an understanding of the relative diversity and abundance of the different species within and between specific environmental samples.

1.8) Comparisons of different tools in other metagenomic pipelines, as well as individual tools that can be used in future studies

In this section, there are various tables that list the different tools that can be used in various stages of the metagenomic analyses of 18S AM fungal data. Even though none of these tools were used in this study, it might be of importance when conducting further studies and designing a pipeline in order to study the biodiversity of 18S AM fungal data.

Table 1.1: Various different tools that can be used to process sequences resultant from high-throughput sequencing. The tools for determining the sequencing quality and the sequence assembly are listed.

Tool	What this tool is used for
Determining DNA sequencing quality	
Phred	It provides base calling, chromatogram display, and high quality sequence region evaluations.
Sequence assembly of the reads	
EGassember	Aligns and merges sequence fragments to reconstruct the original segment or gene (Masoudi-Nejad *et al*., 2006).
CAP3	Very old tool that can be used to develop an understanding of how the processing of sequences work (Huang and Madan, 1999; Huanf and Madan, 1999).
CAP EST Assembler	There is a maximum sequence length that can be used, sequences cannot exceed 30 kb and the maximum number of sequences cannot exceed 10 kb.
Divide-and-Conquer Multiple Sequence Alignment	This tool can be used to do multiple sequence alignments with and determining where conserved regions are present within sequences.

Table 1.2: The various tools available that can deal with sequencing errors. Sequencing errors occur when the DNA sequence does not match the expected DNA/protein sequence. The errors can be checked with GenWise (EMBL-EBI) which compares the sequence against that of a know DNA/protein sequence to a genomic DNA sequence. Sequencing errors allow for introns and frameshift errors to occur.

Tool	What this tool is used for
FrameD	Used to determine where the sequencing errors are (Schiex *et al*., 2003)
Shift	This is a web server used for hidden stops in the frameshift translation.
Path	Protein back-translation and alignment. It addresses the problem of finding distant homologies where the divergence is the result of the frameshift mutations and substitutions (Gîrdea *et al*., 2010).
In-silico.com	Dr. Joseba Bikandi and co-workers developed this tool at the University of the Basque Country. This tool allows in silico experiments, including theoretical PCR amplifications.

Table 1.3: Some of the tools available for genome annotations. These tools can be used for genome annotations of the DNA sequences produced during high-throughput sequencing.

Tool	What this tool is used for
MAKER Web Annotation Service (MWAS)	Easily configurable web-based genome annotation pipeline. It allows the annotation and analysis of small to intermediate amounts of eukaryotic genome sequences to produce an output that can be loaded into a genome database (Holt and Yandell, 2011).
Genome Sequence Annotation Server (GenSAS) **http://gensas.bioinfo.wsu.edu**	It is a website that runs multiple structural and functional annotation tools. It enables visualization and manual curation of the genome sequences. Gene prediction programs can be run as well as homology searches, map ESTs, identify repeats, etc. Each analysis leads to a gff3 file that can be uploaded to a browser like GBrowse.
Genome Annotation Transfer Utility (GATU)	It annotates a genome based on very closely related reference genomes (Tcherepanov *et al*., 2006).
BioGPS	This tool was developed by Scripps Research Institute and can be used as a gene annotation portal.
Riboswitch Explorer (RibEx)	This tool can scan 40 kb of DNA for potential genes that are linked to BLASTP and regulatory elements that include riboswitches. It also permits BLAST analysis (Arbreu-Goodger and Merino, 2005).
FancyGene	It is a web-based tool used to produce images of one or more genes directly on the corresponding genomic locus. It can start with a variety of input formats and rebuilds the basic components of a gene. Users can also superimpose additional features, as biological markers, in specific positions (Rambaldi and Ciccarelli, 2009).
Metagenome Rapid Annotation using Subsystem Technology (MG-RAST)	This is a fully automated tool for annotating samples and provides the annotation of sequence fragments, phylogenetic classification and initial metabolic reconstruction. It also provides functions to compare phylogenetic classifications and metabolic reconstructions of metagenomes (Meyer *et al*., 2008).

Table 1.4: Various tools are available in order to do corrections to genome annotations. Only one tool is listed below, because it was the only tool that seemed to be able to work with the specific data (18S AM fungal).

Tool	What this tool is used for

gbk2tbl	Tbl2asn is a command-line program used for the automation of the creation of sequence records for submission to GenBank. This tool is not user-friendly. Gbk2tbl will generate tables that contain the genome features.

Table 1.5: Tools are also available to create general specialized annotations from the sequencing data.

Tool	What this tool is used for
MultiLocus Sequence Typing (MLST)	This tool works with assembled genomes and contigs (Larsen *et al*., 2012).
KmerFinder	This tools helps to predict the species from which a partial or complete genome is from (Hasman *et al*., 2013).

Table 1.6: An interesting analysis that can be done, is to look at the differences in the genomes and compare the differences regarding the various species within the samples. Tools that can be used to do genome comparisons, are listed below.

Tool	What this tool is used for
Cinteny Server	This tool can be used for synteny identification and analysis of genome rearrangements. It was developed by Sinha and Meller at the Univeristy of Cincinnati and can be used to find similar regions across multiple genomes and measure the extent of genome rearrangements by using reversal distances.
AutoGRAPH	This is an integrated web server that can be used for multi-species comparative genomic analysis. It can be used to construct and visualize similarities on maps between two or three species. It can also be used to highlight evolutionary breakpoints. The web server uses pairwise comparisons of the marker genes (in this case it would be the 18S genes) against a reference gene (Derrien *et al*., 2007).
Kablammo	This tool creates an interactive visualization of BLAST results. It is very user-friendly (Wintersinger *et al*., 2015).
WebACT	This is a web version of ACT (Artemis Comparison Tool) that compares DNA sequences based on Artemis (Abbott *et al*., 2005).
PARIGA	It enables the user to perform BLAST searches on two sets of sequences selected by the user. It stores the two BLAST results in Python object databases and the results can then be filtered according to several

	parameters in real-time fashion without the need to re-run the process (Orsini *et al*., 2013).
OrthoVenn	This is a web server used for genome wide comparisons and annotation of orthologous clusters across multiple different species. Interactive Venn diagrams, summary counts and functional summaries of the clusters shared between species, are displayed (Yang *et al*., 2015).

Table 1.7: There are various tools that can be used to determine phylogeny. Two of them are listed below.

Tool	What this tool is used for
Average Nucleotide Identity (ANI) calculator	This calculator estimates the average nucleotide identity using both the best hits and reciprocal best hits between the two datasets. This tool supports complete and draft datasets (Goris *et al*., 2007).
POGO-DB	This tool is based on computationally intensive whole-genome BLAST analysis and provides various different metrices on pairwise genomes (Lan *et al*., 2014).

Table 1.8: Genome visualization tools can also come in handy when viewing and comparing DNA sequences from different samples. Tools that can be used for genome visualization are listed below.

Tool	What this tool is used for
Gene Structure Display Server (GSDS)	This tool was designed to visualize gene features such as the composition and position of exons, introns and conserved elements. A high-quality image can be generated where the shape and colour of the features can be customized by the user and modified functions on figures are provided. A phylogenetic tree can also be uploaded and added to the generated figure (Hu *et al*., 2015).
GCView Server	This tool is used for comparative metagenomics. Sequence feature information can be visualized and it uses BLAST to compare the sequences to the comparison sequences and then converts the results and any available feature information, or analysis information, into high-quality graphical maps that shows the entire region of interest (Grant and Stothard, 2008).
DNAPlotter	This is an integrative Java application that generate representations of sequences or genomes. It makes use of the Artemis libraries to provide user-friendly usage.

	It filters the features of interest to display on user-definable tracks and produces publication quality images (Carver *et al*., 2008).

Table 1.9: Some tools are needed to just view the sequences and view features and primers within the sequences. One such tool is listed below with a short explanation.

Tool	What this tool is used for
DNAATLAS	The sequences can be imported in any text-based format, parsed and uploaded. This tool infers whether each sequence is a feature, construct, primer, DNA or amino acid. The user can keep track of all the sequences uploaded, as well as the features and primers. The uploaded data can be categorised by using tags. This site requires registration (Ulster Med, 2015).

Table 1.10: The pipelines and their tools used during this analysis to analyse the Roche 454 data of the Rooibos and Honybush samples.

Pipeline	Tool	What this tool is used for
QIIME	Python scripts from QIIME website (http://qiime.org/1.9.0/scripts/index.html)	These scripts are used on the command line in order to do initial processing of the data, picking of OTU's and statistical analysis and phylogenetic tree generation.
RDP	Initial process	Processing of sequence libraries or assemble paired-end reads.
	Classifier	Assign bacterial or archaeal 16S or fungal 28S rRNA sequences to the RDP taxonomic hierarchy.
	Aligner	Alignments of the sequences using the fast, secondary-structure aware Infernal aligner.
	Complete linkage clustering	Clustering the sequences by using the complete-linkage

		clustering method.
	Chimera check using UCHIME	Checking the amplicon sequence files for the presence of chimeras using USEARCH 6.0.
Mothur	Python scripts from the Mothur website (http://www.mothur.org/wiki/454_SOP)	These scripts are used on the command line in order to do initial processing of the data, picking of OTU's and statistical analysis and phylogenetic tree generation.

1.9) Problem statement

According to Rosendahl (2008), many recent studies have focussed on the distribution of AM fungi along varying environmental gradients and the influence on plant communities. Fungal species can be identified from their DNA sequences. A consensus is thus not enough to understand these fungi, and more information is needed on the taxonomy and methods of recognition as well as the concepts of abundance and diversity. There is an association between AM fungi and the majority of crop plants, grasses, trees and shrubs, and ornamental plants (Smith and Read, 2008). In recent years, there has been a growing need to record the biodiversity of these critical soil fungi. Studying AM fungi is a challenge due to their biotrophic nature and inability to be grown in culture. DNA extracted from the Rooibos and Honeybush soil samples were PCR amplified with SSU primers and the AM fungi community was sequenced using 454 FLX pyrosequencing. Attempts to analyze this data and characterize the biodiversity using the MaarjAM database have been hampered as there was no suitable pipeline for classifying and sorting this sequence data. The goal of this study was to compare suitable pipelines that can be used to identify and do statistical analysis on AM samples.

1.10) Aims

DNA extracted from the Rooibos and Honeybush soil samples were PCR amplified in 2014 at the Department of Microbiology and Biochemistry at Rhodes University, with SSU primers and the AM fungi community was sequenced using 454 FLX pyrosequencing. Attempts to analyze this data and characterize the biodiversity using the MaarjAM database have been hampered as there was no suitable pipeline for classifying and sorting this sequence data. This

project aimed to study the different pipelines currently available to analyse 18S AM fungal data. The pipelines and databases studied during this project would in future be applied to more efficiently analyse the 18S sequencing data from AM fungi.

The pipelines focused on during this study were QIIME, Mothur and RDP, which were also compared. The databases focused on were Silva 104 and MaarjAM. Various obstacles were encountered during this study regarding these pipelines and databases (as discussed in the following chapters).

It is important to take note that the three main aims of this study was to perform the quality control and initial analysis of raw DNA reads that were generated by Roche 454 technology from AM communities of natural and commercial Rooibos and Honeybush plants; to identify the analytical pipeline that suited best to work with the Roche 454 generated AM metagenomic data; and to identify the fungal species and OTU's present in the sequenced metagenomic community.

1.11) Objectives

1) To identify the pipelines that can possibly be used to analyse 18S AM fungal data.
2) To identify databases that contain sufficient 18S fungal, and specifically 18S AM fungal, data that can be used to analyse the 18S AM fungal data from the four different samples.
3) To do adequate initial processing of the reads in order to remove low quality reads and reads with ambiguous bases.
4) To do adequate primer trimming to remove all the primers from the reads so not to get biases in downstream analysis.
5) To remove sequences shorter than 10 bp from the sample files after primer trimming.
6) To do clustering of the reads that were related and possibly from the same organisms.
7) To do read binning of the reads according to taxonomic resemblance.
8) To do OTU identification and binning due to this identification.
9) To do phylogenetic tree generation of the AM fungi present in the four different samples.

10) To do statistical analysis on the sequences found as outputs from the different pipelines.
11) To do statistical analysis on the OTUs and phylogenetic trees of the four samples to be able to compare the diversity and abundances of the organisms within the samples.
12) To do BLAST analysis on the sequences from all four samples to find possible hits in any of the databases being studied.
13) To do MSA on the sequences from the pipelines that had similar output sequence statistics after being analysed and then determining the sequences that were identical in the alignments.
14) To construct phylogenetic trees of the longest sequences in the four samples to be able to compare the compositions of the samples.
15) To compare the different pipelines studied to determine which pipeline works best when analysing 18S AM fungal sample data.

Chapter 2: Initial Processing of Reads

2.1) Overview

It is essential to preprocess the raw sequencing reads before subjecting these sample reads to further analysis. Preprocessing included the removal of low quality bases, ambiguous bases and adapter regions from the raw reads. After this was done, stitching together of the preprocessed reads were tried by determining paired end reads. Chimeric reads were also

detected by using UCHIME and USEARCH and removed from the sequences (Santamaria *et al*., 2012). Both UCHIME and USEARCH used databases of known non-chimeric sequences. UCHIME is known to work best in *de novo* studies and so this was used as the program of choice. Greengenes was not used in this study because it is known to be used with bacterial genes, but if it was possible to use this program, it would have been very helpful seeing that it provides important filters that keep the local database synchronized with Greengenes (Ludwig *et al*., 2004).

2.2) Introduction

High-throughput sequencing technologies revolutionized studies done on microbial environments. Previous studies, as the Human Microbiome Project (National Institutes of Health Human Microbiome Project Working Group *et al*., 2009) and the US National Ecological Observatory Network (Hopkin, 2006), are currently good guides to use in order to understand the role of microbial diversity in habitats within our bodies and around the world. Pyrosequencing that uses error-correcting and specific barcodes allow many communities to be analysed simultaneously (Hamady *et al*., 2008). The integration of information from thousands of samples started to reveal large-scale patterns that were previously inaccessible when using lower-throughput sequencing methods. A major problem with achieving such insights has been the lack of sufficient software able to handle the large amounts of data and datasets. Tools do exist to perform library demultiplexing and taxonomy assignment (Cole *et al*., 2009; Schloss *et al*., 2009), but tools for further downstream analysis are very scarce.

Three different pipelines were specifically looked at: QIIME, RDP and Mothur. Below, a description of these pipelines, as well as why they were developed and why they were chosen for this study, follow.

2.2.1) QIIME

QIIME stands for ‘quantitative insights into microbial ecology’ (pronounced ‘chime’). QIIME is an open-source software pipeline that was built by using the PyCogent toolkit (Knight *et al*., 2007). This pipeline was designed to address the problem of taking sequencing data from raw sequence datasets and then interpreting this data, whereafter database deposition can be done. QIIME is available at http://qiime.sourceforge.net/ and it supports a wide variety of microbial community analyses and visualisations. These have been important in network analysis, and within and between sample diversity. QIIME can also provide graphical displays that allow the

users to interact with the data and makes extensive use of unit testing in order to ensure accuracy of results. By including this modularity, OTUs (Operational Taxonomic Unit(s)) can be chosen, sequence alignments can be done, phylogenetic trees can be calculated and taxon-based analysis of the diversities within and between the samples can be integrated (Caporaso *et al*., 2010). QIIME scales to millions of sequences and can therefore be used on platforms from laptops to high-performance computing clusters.

2.2.2) Ribosomal Database Project (RDP)

The RDP (Ribosomal Database Project) pipeline can be found at http://rdp.cme.msu.edu/ and it provides aligned and annotated rRNA gene sequence data. It also provides tools to allow the users to analyse their own rRNA gene sequences in the RDP framework (Cole *et al*., 2013).

The RDP pipeline has been utilized in fields that are as diverse as human health, environmental and microbial ecology, nucleic acid chemistry, taxonomy and phylogenetics. RDP originally just contained aligned and annotated collections of bacterial and archaeal SSU rRNA genes, but it now includes a small collection of fungal LSU rRNA genes (Cole *et al*., 2013). The RDP tools include the Classifier and Aligner and these tools have also been updated to work with the new fungal collection.

The use of high-throughput sequencing to characterize environmental microbial populations has lead to an explosion of data in the past several years. As sequencing technologies have improved, the size of the datasets from environmental samples have also increased. RDP (11 release) provided an expanded set of tools that facilitate the sample analysis of the data (Cole *et al*., 2013). This also included single-stranded and paired-end reads. Most tools are also now available as open source packages that can be downloaded and used locally by the user with high-volume needs, or if the user would like to develop custom analysis pipelines.

RDP release 11.1 (October 2013) contained 2 809 406 aligned and annotated bacterial and archaeal SSU rRNA gene sequences and 62 860 fungal LSU rRNA gene sequences. Most of the rRNA gene sequences in the RDP database were incomplete. Most were derived from PCR amplification products, and a small number of older entries were derived from reverse transcriptase sequencing of isolated rRNA. Seeing that PCR amplification made use of primers to conserved regions within the genes, very few of these sequences covered the 3’ and 5’ ends of the genes. A very diverse selection of complete gene sequences derived from genome

sequencing, are available. A small percentage of bacterial and archaeal sequences originated from organisms grown in culture. Roughly 85% of bacterial -, and 97% of archaeal samples were directly from the environmental samples (Cole *et al*. 2013).

2.2.3) Mothur

Mothur was designed to be a software package that allows its users to use a single package to analyze the community of sequence data. It provides a flexible and powerful software package for analyzing data. Mothur can be used to trim, screen and align sequences, afterwhich the distances can be calculated, the sequences aligned to OTUs and the alpha and beta diversity of the samples can be calculated (Scholss *et al*., 2009).

Since the culture-independent framework for sequencing started in 1985, there was an improvement in the ability to sequence the primary phylogenetic markers as the functional genes from many different environments. More than 25 years later there was over 10^6 rRNA gene sequences available that were deposited in databases such as GenBank (Chan *et al*., 2012). The number of sequences continues to double every 15-18 months (http://www.arb-silva.de/news/view/2009/03/27/editorial/). Technologies have enabled the start of the Human Microbiome Project, and the International Census of Marine Microbes (ICoMM; http://icomm.mbl.edu). Because of the great improvement in sequencing technologies, individual studies have shifted from sequencing 1,000 to 10,000 sequences from multiple different samples to easily a million sequences (Chan *et al*., 2012). Because of this increase in sequences, it has become easier to relate changes in the microbial environment with changes in the ecosystems.

Other major advances in the computational tools have also improved the ability to address ecologically relevant questions by using tools such as ARB, DOTUR, SONS, LIBSHUFF, UniFrac, AMOVA and HOMOVA, TreeClimber and rRNA-specific databases. Microbial ecology has progressed to being able to describe different environments. There are various limitations for these tools in that, firstly, although many of them include rRNA-specific databases and online tools like aligners, classifiers and analysis pipelines, they only allow a limited set of generic analyses; secondly, most of the existing software were developed for analyzing 10^2 to 10^4 sequences (Schloss *et al*., 2009). As the number of sequences increased, it has been important for the different tools and pipelines to be refactored to use more efficient algorithms. Software written in scripting languages such as Perl and Python have been useful

for small datasets, but are slow in comparison to scripts written in C and C++.

The consensus of the development of Mothur was that as the sequencing capacity increases and the research questions become more sophisticated, it is critical that the software has to be flexible and easily maintained (Schloss *et al*., 2009).

2.3) Methodology

Using NCBI (http://www.ncbi.nlm.nih.gov/) to search for fungi sequences in various databases is the easiest and quickest way to find resources. There are also tools that are resource specific, and these resources can be accessed from within the Entrez system. These resources include graphical viewers and comparative analysis tools, but none were sufficient for 18S AM fungal data (Robbertse and Tatusova, 2011). Even though Genbank do have data available on fungi, as well as data mining tools, the data that was available regarding the SSU of Glomeromycota was not sufficient. Thus, this lead to the decision being made that it would be better to use the Silva 104 and MaarjAM databases as reference databases for this study. The UNITE database was not considered seeing as the data it contains is only on the LSU rRNA of fungi of which it is mostly the ITS gene. This was not helpful for this study seeing that the gene sequenced in the four different samples were that of the SSU rRNA (18S) gene of the Glomeromycota.

2.3.1) QIIME

The first pipeline studied was the QIIME pipeline. This was done at the University of Cape Town in the CBIO (Computational Biology) Unit, which forms part of the Department of Infectious Diseases, during August 2015. QIIME is usually used by CBIO to study the abundances and diversities of bacterial strains in different samples and environments using 16S genes. In this pipeline there are a few alterations that can be made in order to determine the diversities and abundances of fungi (ITS and 28S) that come from different samples and environments. This is a quick and easy tool to use for determining the quality of 18S fungal genes in samples, but difficulties came in when 18S AM fungal genes were further studied (as will be discussed in Chapter 3).

Most of the QIIME scripts are Python scripts, but this pipeline also used UNIX commands to pass all the scripts. Some options can be given in order to achieve more specific results if it is needed. All the QIIME scripts can be found at http://qiime.org/scripts/. Figure 2.1 below gives an outline of the methodology followed during initial processing using the QIIME pipeline.

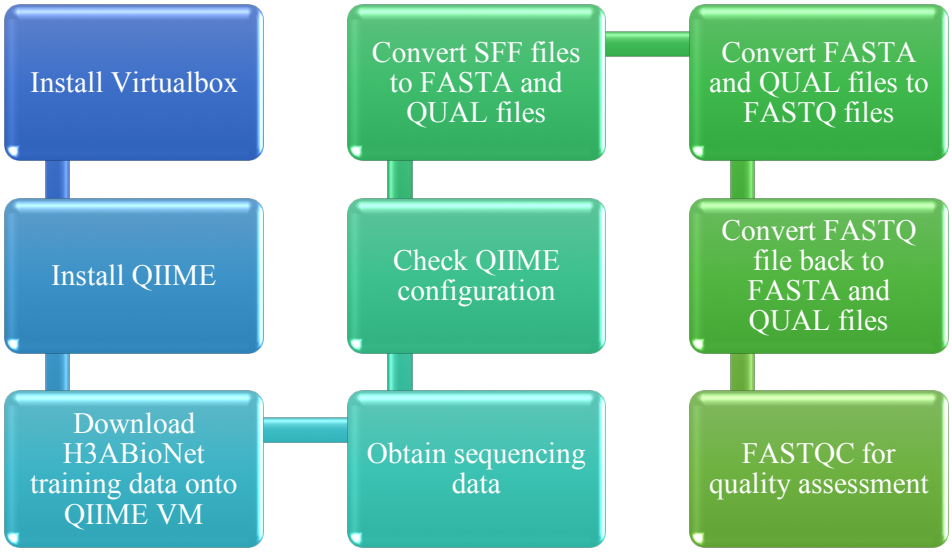


Figure 2.1: An overview of the methodology used to do QC (Quality Control) on the samples. This basic pipeline was used to do QC on both the natural and commercial samples of Rooibos and Honeybush.

2.3.1.1) Virtualbox and QIIME installation

The first step to be able to use the QIIME pipeline, was to install a Virtualbox on the computer being used. The Virtualbox was downloaded from https://www.virtualbox.org/. The Virtualbox was necessary seeing that QIIME works in a UNIX Virtualbox and not on Windows (if using Windows or Mac OS). One of the newer versions of QIIME can be installed on MAC OS 9.0 onwards, but this is not recommended seeing that using QIIME might use a lot of RAM when not placed into a VM (Virtual Machine).

The Virtualbox needs to be correctly installed and running on a UNIX or Windows system before QIIME can be installed. If using MAC OS the process will be the same. QIIME can be downloaded from https://qiime.org and the 'box' created for QIIME was allocated 4096 Mb of RAM. The rest of the installation process was done in the QIIME box by using the instructions given on the QIIME website.

After installing the VM and QIIME, the H3ABioNet training data was installed on the VM (figure 2.2). It was important to get a basic understanding of the QIIME pipeline by first working through the 454 practical that can also be found on the QIIME website. It is important to note that this tutorial is based on 16S data and not 18S data.

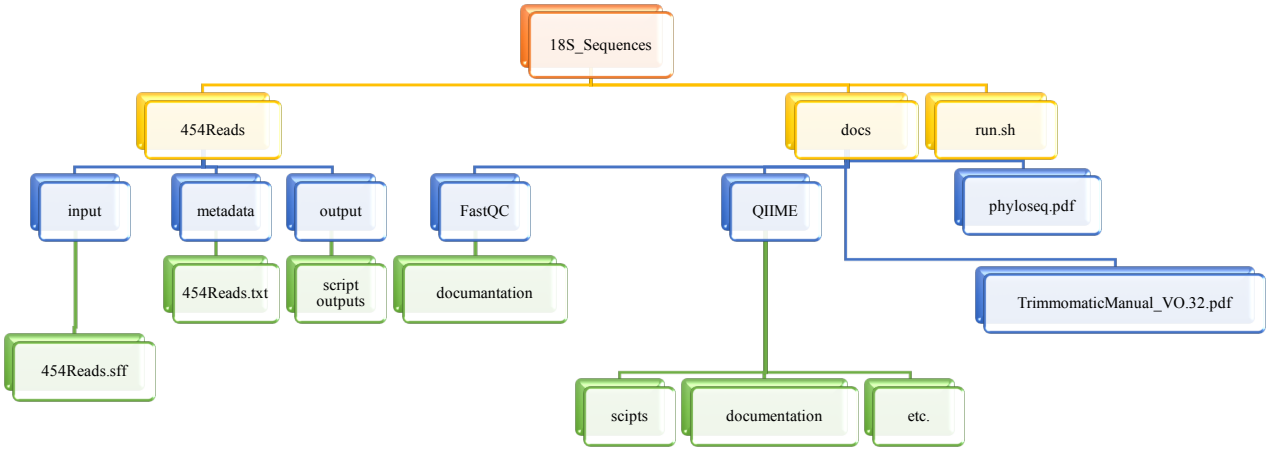


Figure 2.2: Illustration of the directory and file setup needed for the QIIME pipeline to work on the VM. This file setup will usually be correct when downloading the QIIME VM from the QIIME website, but it is necessary to check that all the required files are in all the directories especially when using the training data to familiarize the user with the process.

2.3.1.2) Checking the QIIME configuration

Using the command *print_qiime_config.py [options]* printed the QIIME configuration details and, if there were errors regarding the installation – which there were none of in this case. Table 2.2 shows the different optional tests that can be performed when checking the QIIME installation.

Table 2.2: The different QIIME optional tests that can be run when checking the installation configuration of QIIME. These optional tests will give more detail as to which errors occurred as well as if there is a need to suppress the help details. More detail regarding this can be found on the QIIME website.

[options]	Meaning
-t	**--test** Test the QIIME install and configuration [default: False]
-b	**--qiime_base_install** Suppress help
-f	**--qiime_full_install** If passed, report the dependancies required for the QIIME full install. To perform tests of the QIIME full install, -t must also passed. [default: False]
--haiku	Suppress help

The command passed in this study was print_qiime.config.py to print the basic QIIME configuration details. No errors were observed and analysis could be started.

2.3.1.3) Obtaining sequencing data

The raw sequencing reads from both the natural and commercial samples of Rooibos and Honeybush were obtained by using 454 FLX pyrosequencing with gene specific primers. Table 2.3 below illustrates the primers and tags used to amplify the 18S AM fungal genes. The sample sizes ranged from ~ 10 000 sequences to over 17 000 sequences per sample.

Table 2.3: The primers and tags used to amplify both the natural and commercial samples of Rooibos and Honeybush by the Rhodes Microbiology and Biochemistry Department. The SSU primers were first amplified and then the MID tags were added during a second amplification (nested PCR) to be able to distinguish between the natural and commercial samples.

Sample	Forward Primer (NS31)	Reverse Primer (AML2)	Forward MID tag	Reverse MID tag
Honeybush natural	5'-TTGGAGGGCAAGTCTGGTGCC-3'	5'-GAACCCAAACACTTTGGTTTCC-3'	5'-CGTATCGCCTCCCTCGCGCCATCAGCGTGTCTCTA-3'	5'-CTATGCGCCTTGCCAGCCCGCTCAGCGTGTCTCTA-3'
Honeybush commercial			5'-CGTATCGCCTCCCTCGCGCCATCAGCTCGCGTGTC-3'	5'-CTATGCGCCTTGCCAGCCCGCTCAGCTCGCGTGTC-3'
Rooibos natural			5'-CGTATCGCCTCCCTCGCGCCATCAGTAGTATCAGC-3'	5'-CTATGCGCCTTGCCAGCCCGCTCAGTAGTATCAGC-3'
Rooibos commercial			5'-CGTATCGCCTCCCTCGCGCCATCAGTCTCTATGCG-3'	5'-CTATGCGCCTTGCCAGCCCGCTCAGTCTCTATGCG-3'

The SFF files containing the raw reads of the four samples amplified during 2014 in the Department of Biochemistry and Microbiology were obtained from Dr Gwynneth Matcher. The raw read files were used to be able to do analysis from the start where both the primers and the tags were still present in the files.

Nested PCR was used to amplify the sequences needed for this analysis. Firstly, amplification was done using the gene specific primers for each sample to amplify the 18S AM fungal genes. After the first round of PCR, another round followed in which the MID tags were added to the 18S gene and primer complex to be able to distinguish between the natural and commercial samples.

2.3.1.4) Convert SFF files to FASTA and QUAL files

The script used to complete the task of converting the raw SFF sequencing files to FASTA and QUAL files, were *process_sff.py*. This script was used to convert the directory of SFF files into FASTA, QUAL and flowgram files. This script may also be used with various options in order to achieve certain file conversions, as shown in table 2.4.

Table 2.4: The different compulsory and optional options that can be used when converting SFF files to FASTA and QUAL files using the QIIME pipeline. The input file [-i] is necessary to be able to do the conversion, but other options can be used for personal preference in the newly made FASTA and QUAL files.

Option	Meaning
[REQUIRED]	
-i	**--input_dir** Input directory of SFF files or a single SFF filepath
[OPTIONAL]	
--no_trim	Do not trim sequence/qual (requires -use_sfftools option) [default: False]
-f	**--make_flowgram** Generate a flowgram file [default: False]
--use_sfftools	Use the external programs *sfffile* and *sffinfo* for processing, instead of the equivalent Python implementation
-o	Input directory of SFF files [default: same as input directory]
-t	**--convert_to_FLX** Convert Titanium reads to FLX length [default: False]

The script used resulted in FASTA and QUAL formatted files. The FASTA files contained the sequences, whereas the QUAL files contained the quality scores of the sequences. These quality scores were later used in the QC of the sequences in the QIIME pipeline.

2.3.1.5) FASTQC for quality assessment

FASTQC had to be done in order to determine the quality of the reads before and after primer and tag trimming were done. Before this QC can be done, the FASTA and QUAL files needed to be converted to a FASTQ file. It was important to use the matching FASTA and QUAL files to create the FASTQ file for each sample. The script used to create the FASTQ file was *convert_fastaqual_fastq.py*. From this newly generated FASTQ file a FASTA and QUAL file were again generated. Table 2.5 can be used to give specific options as to which directories the files should be created in.

The reason for creating a FASTQ file is that this file omitted the redundant sequence labels of the quality scores. The quality scores were assumed for each sequence immediately after the sequence with which they were associated, was scored. The output FASTQ file was generated in the specific output directory and was named the same as the input FASTA file. The FASTQ file was split into FASTA and QUAL files also in the specific output directory. The script used was *convert_fastaqual_fastq.py [options]*.

Table 2.5: Various options that can be used when converting the FASTA and QUAL files to a FASTQ file for each sample. There were both required and optional options that could be used with the Python script.

Option	Meaning
[REQUIRED]	
-f	**--fasta_file_path** Input FASTA or FASTQ file
[OPTIONAL]	
-q	**--qual_file_path** Required input QUAL file if converting to FASTQ
-o	**--output_dir** Output directory. Will be created if it does not exist [default: .]
-c	**--conversion_type** Type of conversion: fastaqual_to_fastq or fastq_to_fastaqual [default: fastaqual_to_fastq]
-a	**--ascii_increment** The number to add (substract if converting from FASTQ) to the quality score to get the ASCII character (or numeric quality score) [default: 33]
-F	**--full_fasta_headers** Include full FASTA headers in output file(s) (as opposed to merely the sequence label) [default: False]
-b	**--full_fastq** Include identifiers on quality lines in the FASTQ file (those beginning with a "+"). Irrelevant when converting from FASTQ [default: False]
-m	**--multiple_output_files** Create multiple FASTQ files, one for each sample, or create multiple matching FASTA/QUAL for each sample [default: False]

The outputs were complete FASTQ files for each of the samples. These files omitted the redundant sequence labels on the quality scores and was then again split into matching FASTA and QUAL files for each sample.

2.3.2) RDP pipeline

The RDP pipeline (figure 2.3 and figure 2.4) aimed to simplify the processing of large rRNA sequence libraries that included both single-strand and paired-end reads that were obtained through high-throughput sequencing technologies. The RDP site (http://pyro.cme.msu.edu) offers tools that can be used for assembly, quality filtering, taxonomy based analysis and taxonomy independent analysis tools that can be used to convert data to formats that are suitable for common ecological and statistical packages. For large datasets it will be preferable to use the command line tool which can be found at RDPTools GitHub (https://github.com/rdpstaff). The RDP help pages (http://pyro.cme.msu.edu/pyro/help.jsp) can also be of use and there are procedural tutorials (http://rdp.cme.msu.edu/tutorials/RDPtutorial_CONTENTS.html) that can assist with initial analysis (Cole *et al*., 2009).

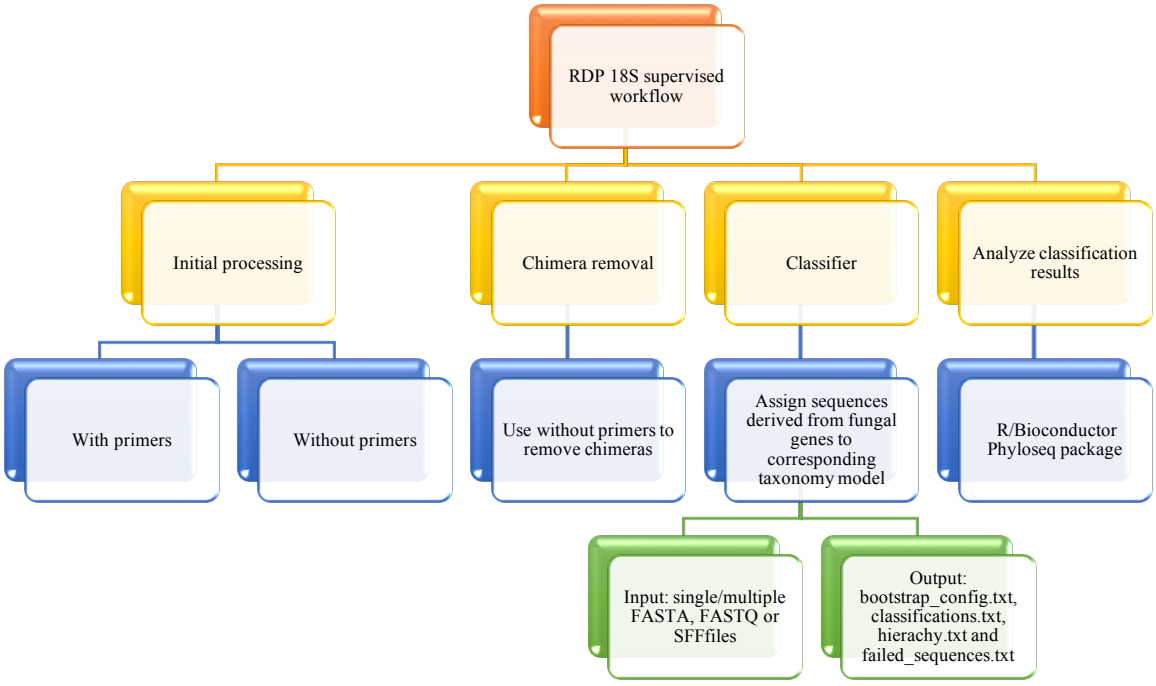


Figure 2.3: Illustration of the RDP standalone procedure. This procedure was used in order to analyse the 18S AM fungal data of both Rooibos and Honeybush commercial and natural samples. Only the initial processing of the reads were discussed in chapter 2. The chimera removal and classification were discussed in chapter 3. A method to analyse the classification results with R and RStudio, was shortly discussed in appendix VII.

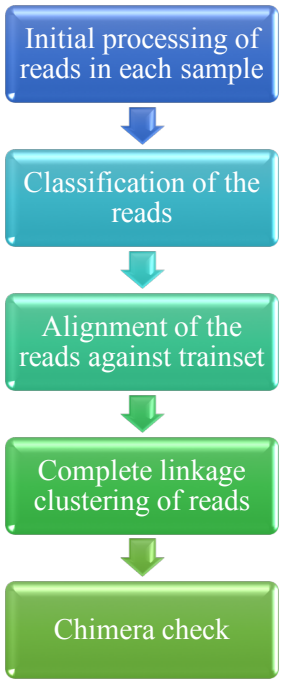


Figure 2.4: The methodology used for the RDP pipeline initial processing of the reads. In depth discussion of the methodology is included in this section. The whole RDP pipeline used in this analysis was used in the GUI mode.

2.3.2.1) Initial processing of reads

The initial processor module is limited to 20 000 000 sequences for each input file. The initial processing steps included the sorting of the raw reads into those from each of the original samples, trimming off the key tag and primers, and removing the sequences that were of low quality. The input file could be a single file or a compressed file containing multiple sequence files. For larger datasets, the command line tools need to be implemented, but this was not necessary seeing that there were at most ~ 17 000 sequences in one sample file. RDP uses a new Assembler of Paired-End reads (APE) that is an extended program based on the original PANDASeq (Masella *et al*., 2012). This program uses a new statistical model to compute quality scores in the overlapping region and to handle more complex overlapping layouts. Reads with a Q-score (quality score) of 25 to 27 was recommended to filter out low community assembled reads based on the results using MiSeq defined datasets (Cole *et al*., 2014).

Initial processing steps included optional sorting of the raw reads into those from each of the original samples, trimming off the primers, and removing sequences that were of low quality. The SFF formatted files were used in order to start with the same files as with the QIIME pipeline. The 18S gene was the ideal gene to select to do sorting against. No tag file was needed seeing that the tags were already previously removed, but both the forward and reverse primers

were specified. For future analysis, it is important to note that the primers cannot have a length of more than 64 bases.

2.3.3) Mothur

Mothur was designed by Schloss *et al*. (2009) at the Department of Microbiology and Immunology at the University of Michigan Medical School. Mothur had designed a tutorial in order to work with fungal 454 sequencing data. This tutorial was found at http://www.bio.utk.edu/fesin/FESIN2010/Workshop2010/Amend/Mothur_tutorial.pdf.

Three files were needed for this analysis: the sequences FASTA files, the QUAL files and the OLIGOS files. The FASTA files had to contain the sequences in FASTA format; QUAL files, the quality scores for each base call; and the OLIGOS file had to contain the primers used for each sample. The FASTA and QUAL files were outputs from the QIIME pipeline initial analysis (see 2.3.1.4).

2.3.3.1) Initial processing and clean-up of sequences

The code used in this section is discussed in appendix II.

2.3.3.1.1) summary.seqs

Firstly a summary of the sequences were made with the **summary.seqs** command. In this summary was found how many sequences were present per sample, the lengths of the shortest and longest reads, and the median amount of base pairs, the amount of ambiguous base calls (Ns) and the amount of homopolymeric (single nucleotide repeat) sequences.

2.3.3.1.2) trim.seqs

The sequences were then cleaned up by using the **trim.seqs** command. This command provided preprocessing features to screen and sort pyrosequences. Similar analysis was provided in the RDP pipeline, but this step had added flexibility and speed, and could use the barcode information to generate group files and split the fasta file into sub-files (a similar step is also used in the QIIME pipeline), screened sequences based on the QUAL file that came from 454 sequences, sifted the sequences based on sequence length and the presence of ambiguous bases, and also got the reverse complement of the sequences present within the sample files. This sequence analysis was geared towards pyrosequencing collections, it could also be used with traditional Sanger sequences.

The barcodes with the primers could be used to multiplex a number of samples in a single run, but this was not necessary seeing that all the samples were already separated. The OLIGOS file was created in order for the program to determine which sequences go with which samples. The "oligos" option took a file that contained the sequences of the forward and reverse primers with their sample identifiers. The forward and reverse primers were the same for each sample, but the identifiers (or MID tags) differed. Each line of the OLIGOS file started with the key words "forward" and "reverse". The line could also start with a "#" in order to tell Mothur to ignore that line of the file. No "#" was used in the OLIGOS file in order to allow Mothur to screen for both the forward and reverse primers because some of the sequences generated by the 454 FLX pyrosequencing amplification were longer than 500 bp. The oligos can be added to the file in upper or lower case letters. It has been shown that sequencing errors in the PCR primer region of a sequences correlate highly with poor sequence quality (Mothur 18S tutorial). Mothur will therefore only allow an exact match to the primer sequences that were provided.

Three new files were created in the working directory: a *scrap.fasta* that contained the sequences that did not pass, *trim.fasta* that contained the sequences that did pass, and a *trim.fasta.summary* that contained a sequence-by-sequence analysis of what failed/passed. From the *trim.fasta* file a summary was again made with the **summary.seqs** command in order to see the statistical results of the sequences after doing processing.

The sequences could also be trimmed based on the quality scores. The 454 platform could also create a QUAL file (quality file) that mirrors the FASTA file. The difference between these two files are, in place of letters in the FASTA file, the QUAL file has numbers that indicate the quality of each base call. An ambiguous base was represented with a "0" because it had a poor base call. If a qfile was provided, it will also be necessary to provide either the Qaverage or the Qthreshold options as well. Mothur has a number of other quality-checking options:

- Qaverage is the average Q-score of a sequence,
- Qthreshold is the threshold Q-score of a sequence,
- Qtrim is to trim away the low quality sections of a sequence,
- Maxambig is the number of allowable Ns,
- Maxhomop is the length of the tolerable homopolymers,
- Minlength is the minimum length of the sequences, and
- Maxlength is the maximum length of the sequences.

All the above quality checks can be combined into one command and changed as seen fit. A moderately tolerant quality check was used:

```
MOTHUR > trim.seqs(fasta=SSU.fna, qfile=SSU.qual, oligos=SSUoligos,
minlength=300, maxlength=500, maxambig=0, maxhomop=10, qaverage=25)
```

The summary.seqs command was used again in order to see the amount of the original sequences that passed.

2.3.4) Comparisons of the initial processing results

After the initial processing of the four samples' reads were done with QIIME, RDP and Mothur, the sample files were compared. This was done by using the summary.seqs command from Mothur. This was used because it gave a quick statistical look at the sequences within each sample file after initial processing was done. These results were then used to compare the pipelines to get an idea as to which of the three pipelines gave more or less the same results.

Another method of comparing the results of initial processing was to run the code found in Appendix V. This code was run after the Mothur comparisons and also during the alternative method discussed in Appendix V. This code is a Python script used to determine the sequence identities of the longest reads.

2.3.5) NextGen Workbench analysis

Procedures done in NextGen Workbench and after, are discussed in Appendix III. This is an alternative method that was tried. It is not a separate pipeline, and thus not discussed in the Introduction section (section 2.2), but rather an alternative program that can be used when doing the initial processing of the reads.

2.4) Results and Discussion

The initial sample sizes that were analysed were between ~ 10 000 reads and ~ 17 000 reads per sample. The samples were Honeybush natural, Honeybush commercial, Rooibos natural and Rooibos commercial. This section discusses the results of the initial processing of the reads in which the primers were trimmed and quality control was done by using the QIIME, Mothur and RDP pipelines. At the end of the discussion, there is a comparison of the different sample

reads before and after initial processing was conducted. This comparison gave an idea as to which pipeline(s) were more compatible to use when doing initial processing on AM fungal samples.

2.4.1) QIIME

2.4.1.1) FASTQC results

Analysis of the results were done while comparing the results from this study against that of the ideal results found at the website that regarded QIIME FASTQC results (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/). The results compared well with the ideal results, but it needed to be kept in mind that the results on the above website was that of a 16S sample and not 18S AM fungi.

Adapter content

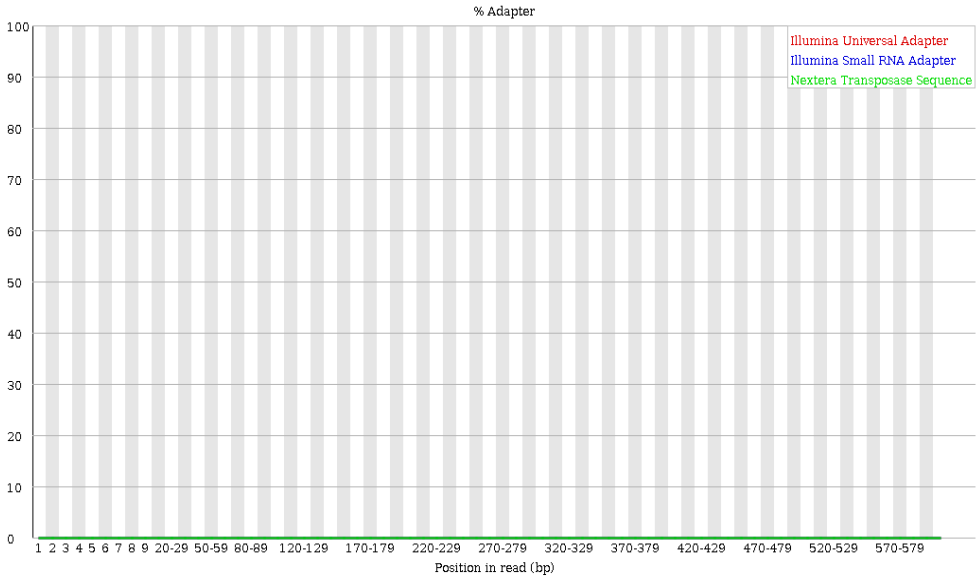


Figure 2.5: Adapter content of the Honeybush natural sample. This figure showed that no MID tags were present in any of the reads.

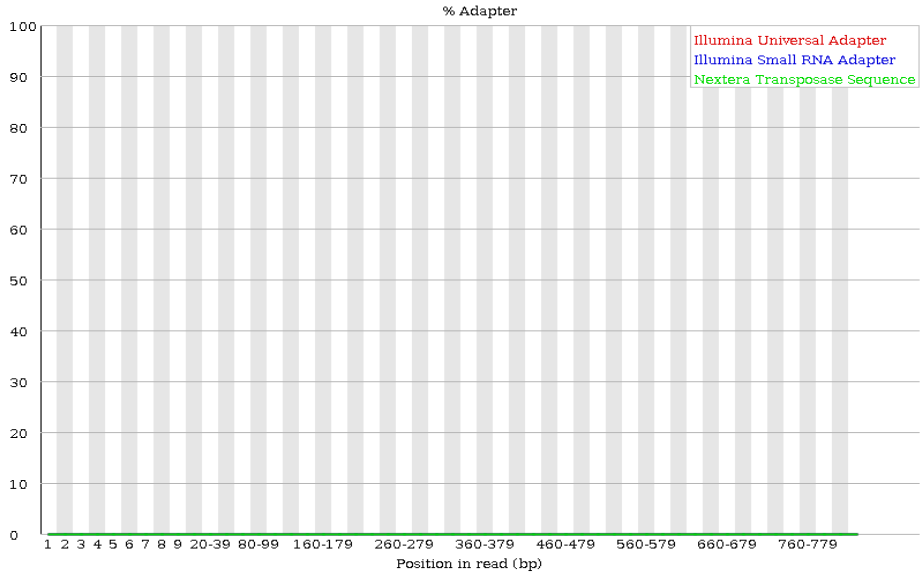


Figure 2.6: Adapter content of the Honeybush commercial sample. This figure showed that there we no MID tags present in the sample.

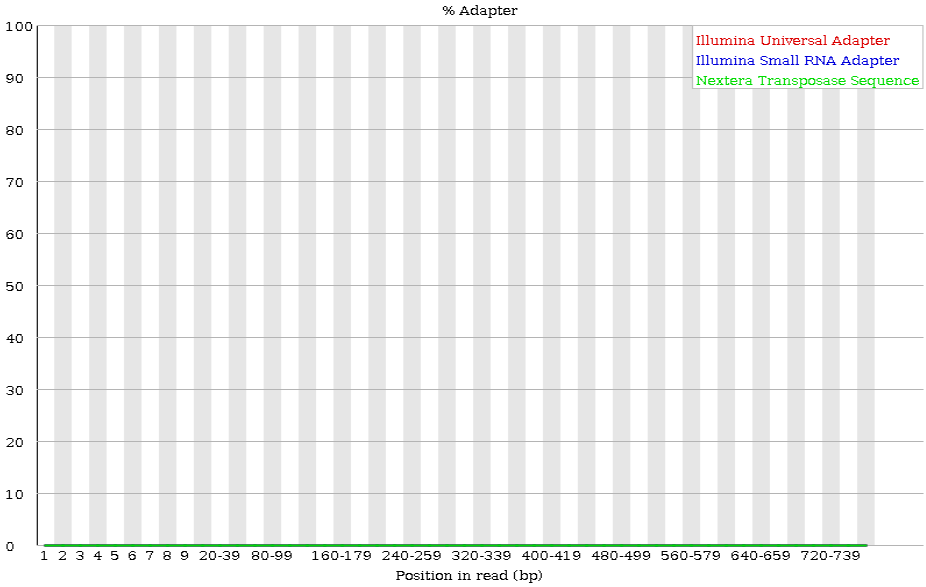


Figure 2.7: Adapter content of the Rooibos natural sample. This sample contained no MID tags in the sequences.

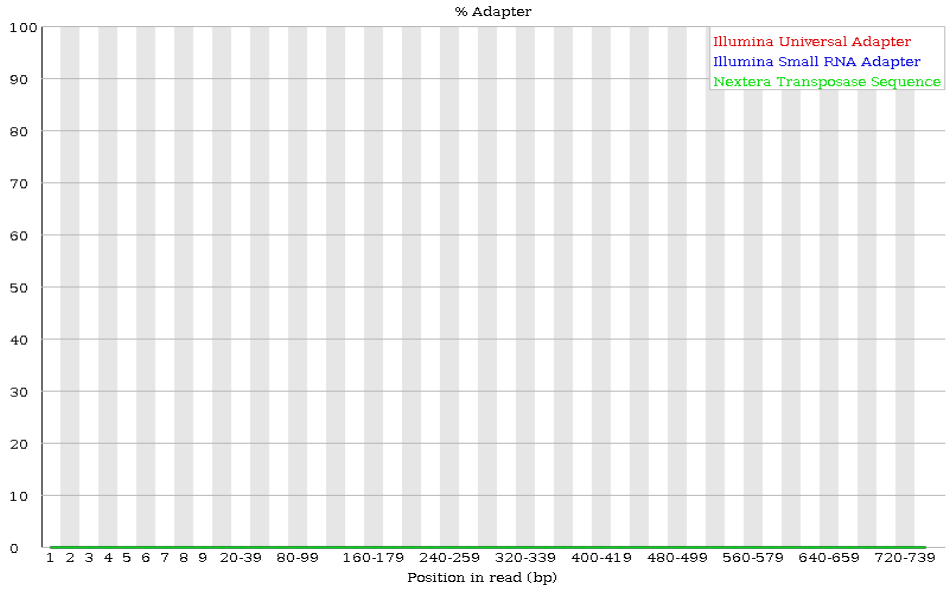


Figure 2.8: Adapter content of the Rooibos commercial sample. This figure indicated that no MID tags were present in the sample.

Figure 2.5-2.8 showed the adapter/tag content of each individual sample. In each of these samples there were no MID tags present in the sample sequences and so no alternative trimming was neccessary. These tags were taken out by Dr Gwynneth Matcher (from the Department of Microbiology and Biochemistry at Rhodes University) right after the PCR reaction was completed and thus the raw SFF files did not contain any tags. The Roche platform that was used for PCR amplification, comes with an SFF tool that automatically removed the MID tags when the samples were split into natural and commercial samples according to their MID tags.

Removal of duplicated sequences

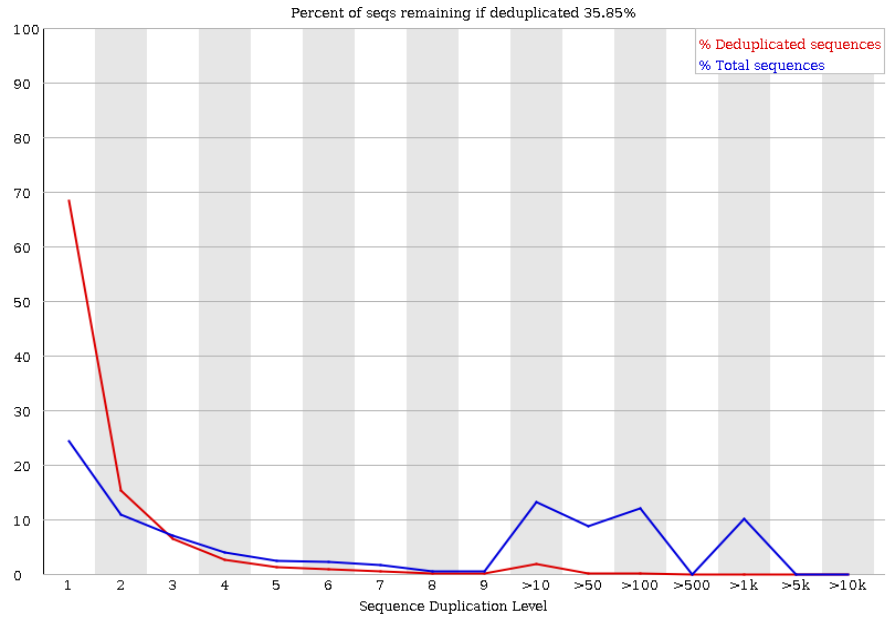


Figure 2.9: The percentage deduplicated sequences (red) and percentage total sequences (blue) in the Honeybush natural sample. This figure showed that duplicates were taken out of the sample during FASTQC.

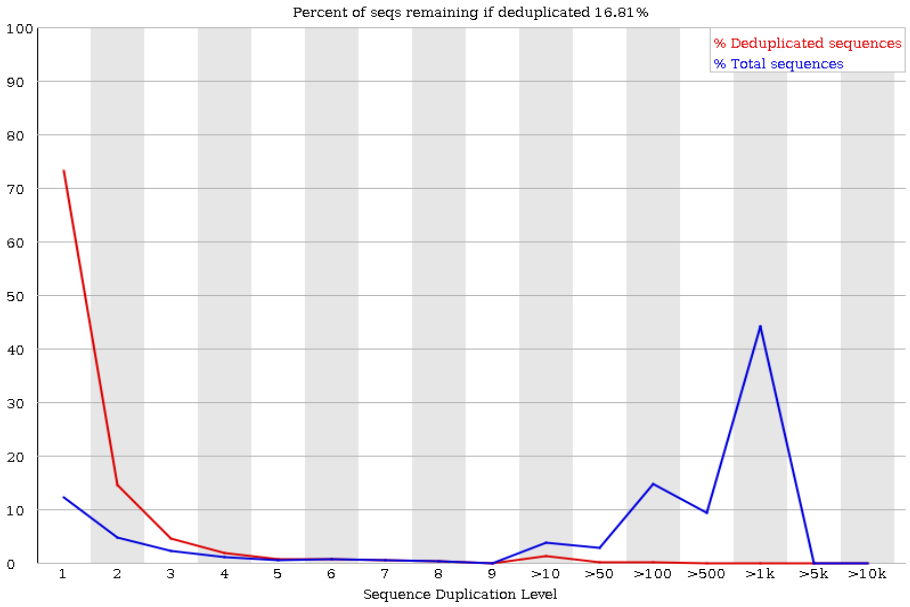


Figure 2.10: The percentage deduplicated sequences (red) and percentage total sequences (blue) in the Honeybush commercial sample. This figure showed that duplicates were taken out of the sample during initial processing.

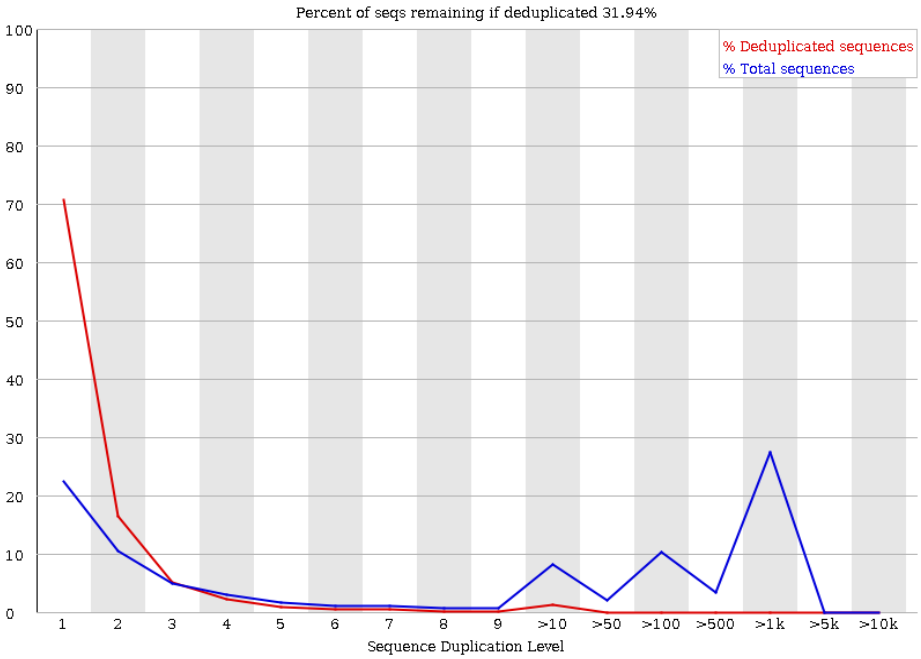


Figure 2.11: The percentage deduplicated sequences (red) and percentage total sequences (blue) in the Rooibos natural sample. This figure showed that duplicates were taken out of the Rooibos natural sample during initial processing.

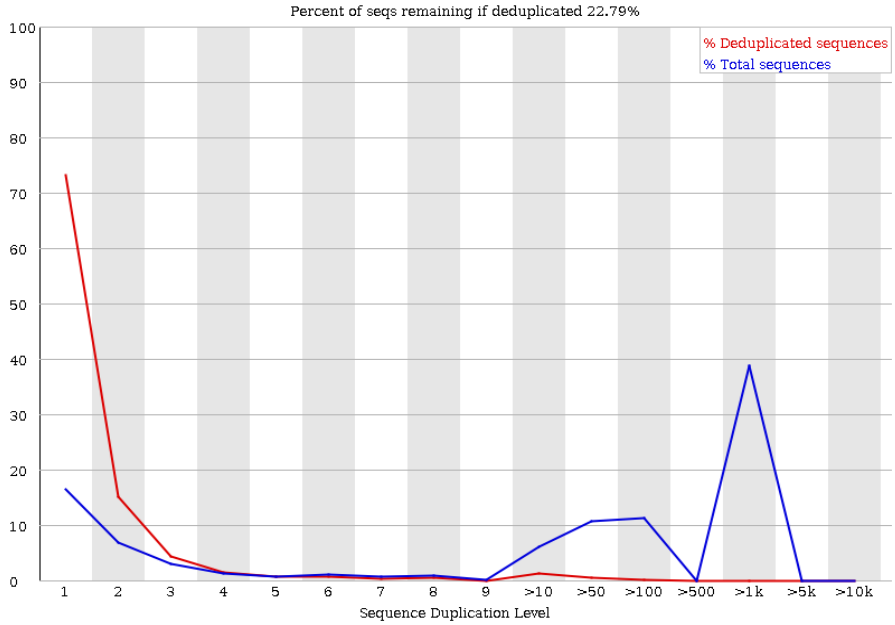


Figure 2.12: The percentage deduplicated sequences (red) and percentage total sequences (blue) in the Rooibos commercial sample. This figure showed that duplicates were taken out of the Rooibos commercial sample during initial processing.

In the different sample libraries that contained different sequences, most of the sequences occurred only once in the final sample set. But there was a possibility that there would have been a low level of duplication. This duplication was indicated by the red lines in figure 2.9-2.12. This red line indicated a very high level of coverage of the target sequence. The

duplication that was shown in blue was more likely to indicate a kind of enrichment that occurred during PCR amplification. This was an enrichment bias within the sequences.

Figures 2.9-2.12 showed that this module counted the degrees of duplication for each of the sequences in the library and then it created plots that illustrated the relative number of sequences that had different duplication levels.

When in future studies it will be possible to work with larger datasets, it is possible to cut down the memory requirements for this module by using only the sequences that first appear in the first 100 000 sequences in each of the sample files being analysed. The 100 000 sequences will be enough to get a good idea of the duplication level of the whole sample file. Each sequence in the sample file can then be tracked to the end of the file in order to get a good representative count of the overall duplication level. Then, in order to cut down on the amount of information that will be on the plot, any of the sequences that had more than 10 duplicates can be placed into group bins that give a clear impression of the overall duplication level without having to show the individual duplication values for the sequences (Caporaso *et al*., 2010).

This module required exact sequence matches over the whole length of the sequences. Any reads that were longer than 75 bp in length were truncated to 50 bp in length. The reason for this was because the reads may have contained more sequencing errors which could have caused an increase in the observed diversity in later analysis. This would have tended to underrepresent the highly duplicated sequences.

The duplication plots in figures 2.9-2.12 showed the proportion of the entire sequence library that was made up of sequences in each of the different duplication bins. The blue line showed the full sequence set and how the duplication levels were distributed. The red line showed the sequences which were de-duplicated, and the proportions shown were the proportions of the de-duplicated set which came from the different duplication levels in the original sample data.

In a library that has very diverse sequences, the sequences would have fallen in the far left of the plot in both the red and blue lines. But if there is a general level of enrichment, that indicates broad oversequencing in the library, the plot will tend to flatten the lines and lower the low end and raise the other categories. This was thought to be represented by this case. When there

was a more specific enrichment of the subsets and possible presence of low complexity reads, the plot will tend to spike towards the right of the plot. This was thought to be the case with all four samples. But when there was a high duplication peak that appeared in red, it made it a high proportion of the original sample library. It usually disappeared in the blue line as it made up a very small proportion of the de-duplicated set. When peaks persisted in the blue line then it was an indication of a large number of the different highly duplicated sequences. This could either have indicated a contaminant set or a very severe technical duplication (http://www.bioinformatics.babraham.ac.uk/).

This module was also used to determine if there were an unexpected loss of sequences when de-duplication took place. During the analysis of the overrepresented sequences, this module spotted an increase in exact duplicated sequences. There was a different subset of problems that may have caused the results to be less reliable. Firstly, the presence of very long sequences in the sample that have poor quality because of random sequencing errors, will reduce the counts for exact duplicates dramatically. Secondly, partial sequences within the samples which appeared at a variety of places within the sequences was not seen by the per base content plot or by the duplicated sequence analysis.

Kmer analysis

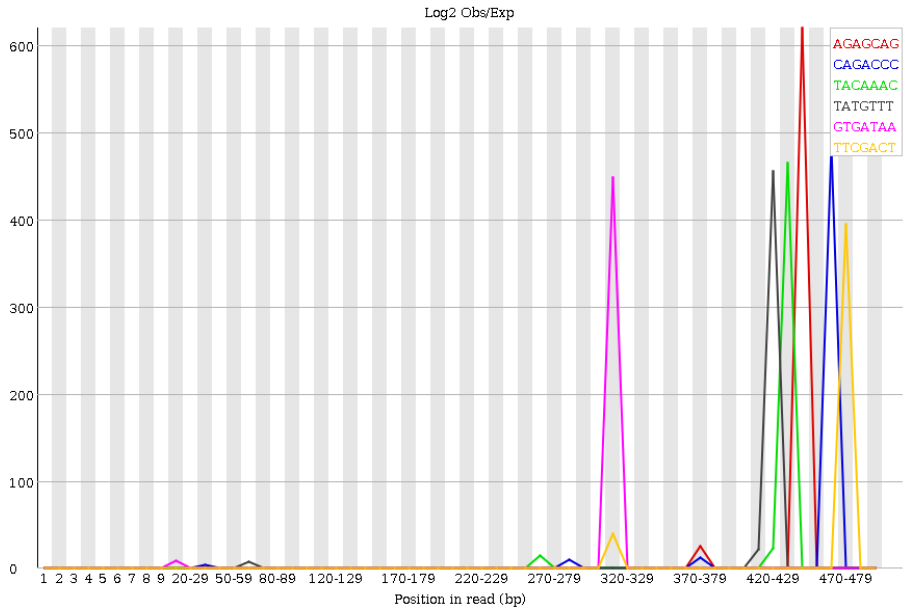


Figure 2.13: The Kmer profiles of various Kmers selected by default in the Honeybush natural sample by FASTQC. The selected Kmers were mostly prevalent in the longer sequences in the sample.

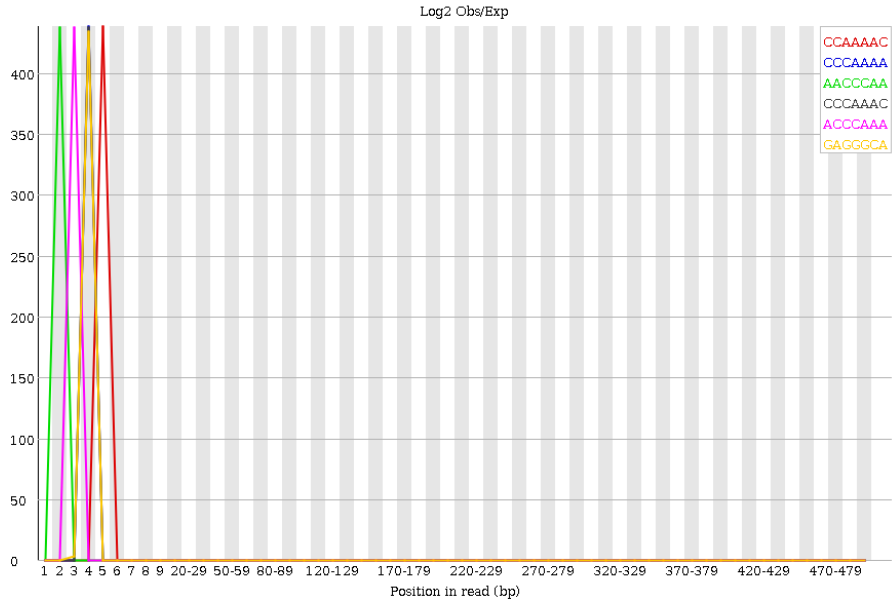


Figure 2.14: The Kmer profiles of various Kmers in the Honeybush commercial sample selected by default by FASTQC. The selected Kmers were mostly prevalent in the shorter sequences in the sample.

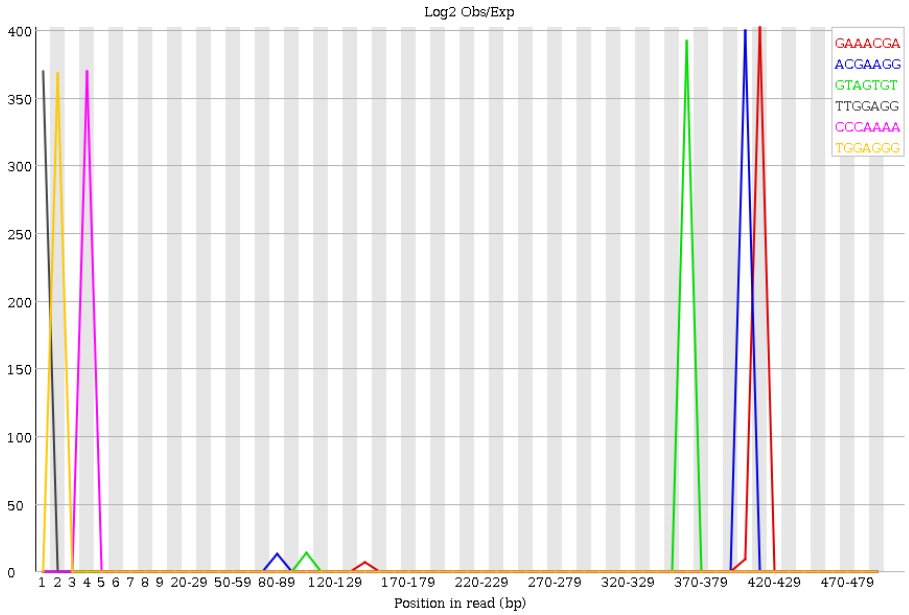


Figure 2.15: The Kmer profiles of various Kmers in the Rooibos natural sample selected by default by FASTQC. The selected Kmers were mostly prevalent in the shorter sequences in the sample, but some were present in longer sequences.

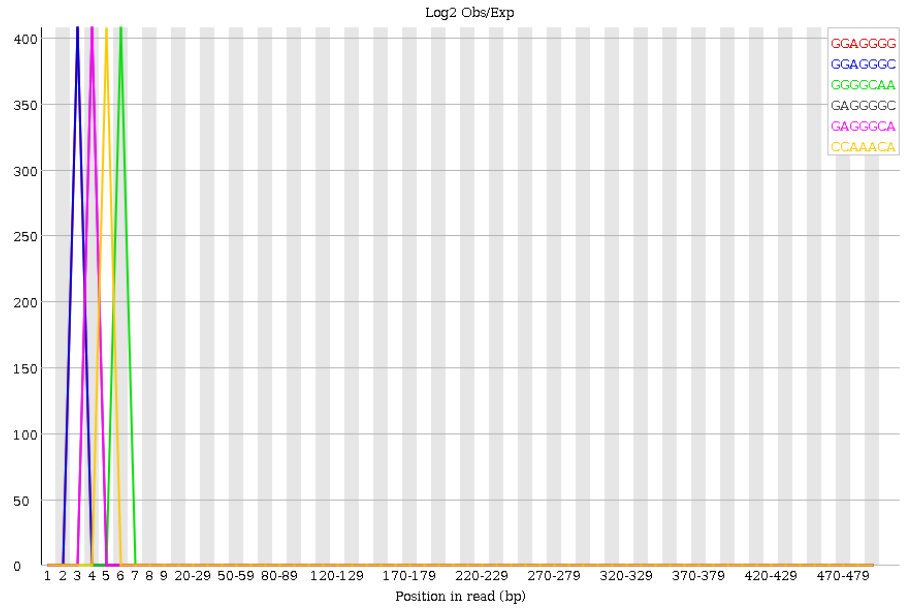


Figure 2.16: The Kmer profiles of various Kmers in the Rooibos commercial sample selected by default by FASTQC. The selected Kmers were mostly prevalent in the shorter sequences in the sample.

During the analysis of the Kmer content, the module did generic analysis of the Kmers in the sample in order to find those Kmers which did not have an even coverage through the length of the reads within the samples. This then lead to finding a number of different sources of bias within the samples. The biases included in the Kmer profiles were the presence of tag sequences that built up on the end of the sequences. The presence of some overrepresented sequences in the samples caused the Kmer content plot to be dominated by the Kmers the sequences contained. It is usually difficult to recognize any other biases that may be present within the samples when the Kmer plot is dominated by Kmers that the sequences contain.

The class of sequences that were analysed were the MID tags and primers. It was important to know whether the sequences contained significant amounts of tags and primers and if it was necessary to trim the adapter sequences or not. The primer sequences that were still attached to the sequences could have caused problems in further downstream analysis. Kmer analysis spotted this kind of contamination, but it was not always clear.

This module did a specific search for a set of separately defined Kmers and then gave a view of the total proportion of the sample which contained these Kmers. The resultant trace was generated for all the primers in the tag and primer configuration file so the tag and primer content of the library was clearly seen (Caporaso *et al.*, 2010).

Figures 2.13-2.16 showed the cumulative percentages of counts of the proportion of the samples which had each of the primer sequences at each position. Once a sequence was seen in a read, it was counted as being present right through to the end of the tag and so the percentages on the graph increased as the read length went on.

The Kmer module was based on the assumption that small fragments of sequences should not have positional bias when it appears in a diverse sample. There are some biological reasons why Kmers could get enriched or depleted overall, but these biases will then affect all the positions within the sequences equally. This module measured the number of each 7-mer at each position in the samples and then used a binomial test to look for significant deviations from an even coverage at all the different positions within the sequences. Kmers that were a positional bias enrichment was reported and the 6 top most biased Kmers were additionally plotted to show their distribution (Caporaso *et al*., 2010).

In order for this Kmer module to run in a reasonable time, 2% of the whole sample was analysed and the results were extrapolated to the rest of the sample reads. The sequences that were longer than 500 bp was truncated to 500 bp for this analysis.

Per base N content

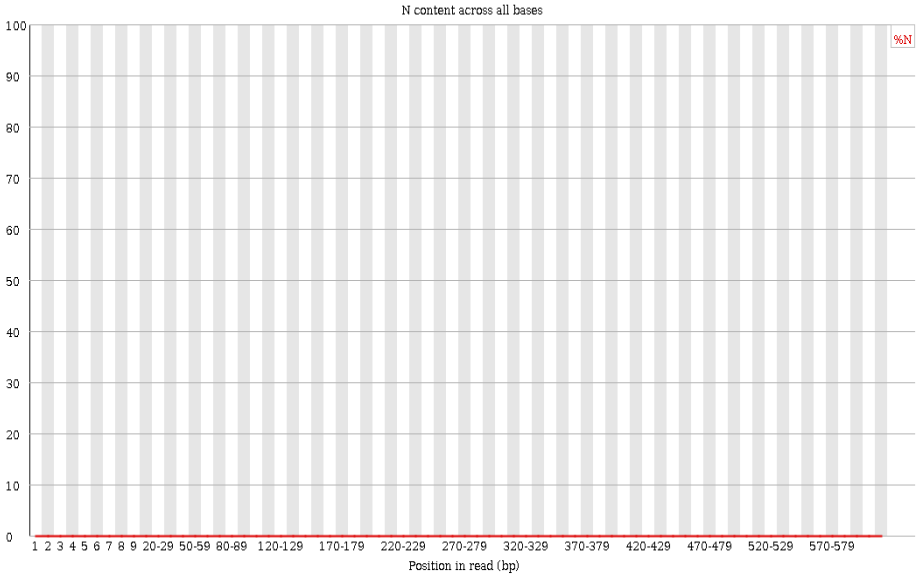


Figure 2.17: The per base N content within the Honeybush natural sample. This figure showed that there were no ambiguous bases present within the sample.

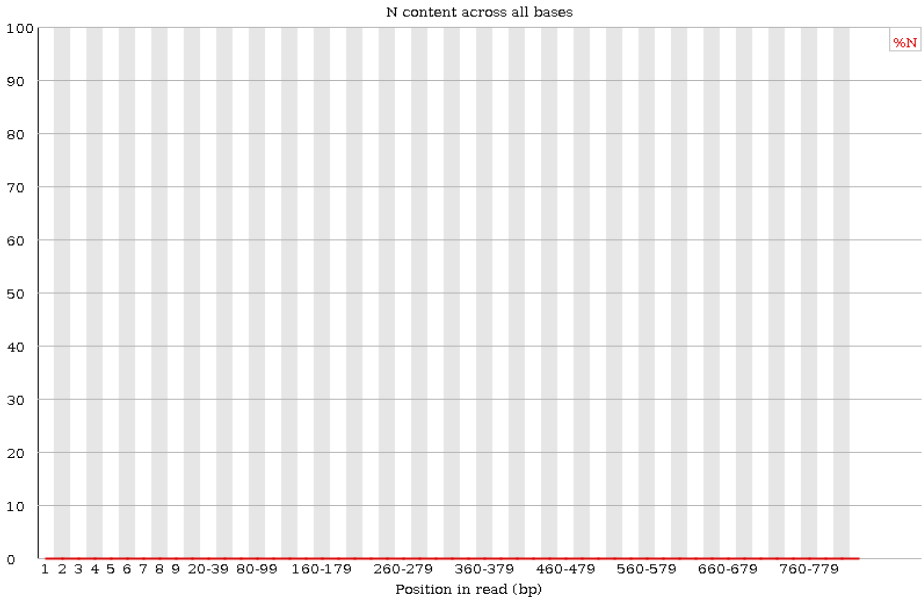


Figure 2.18: The per base N content of the Honeybush commercial sample. This figure showed that there were no ambiguous bases present within this sample.

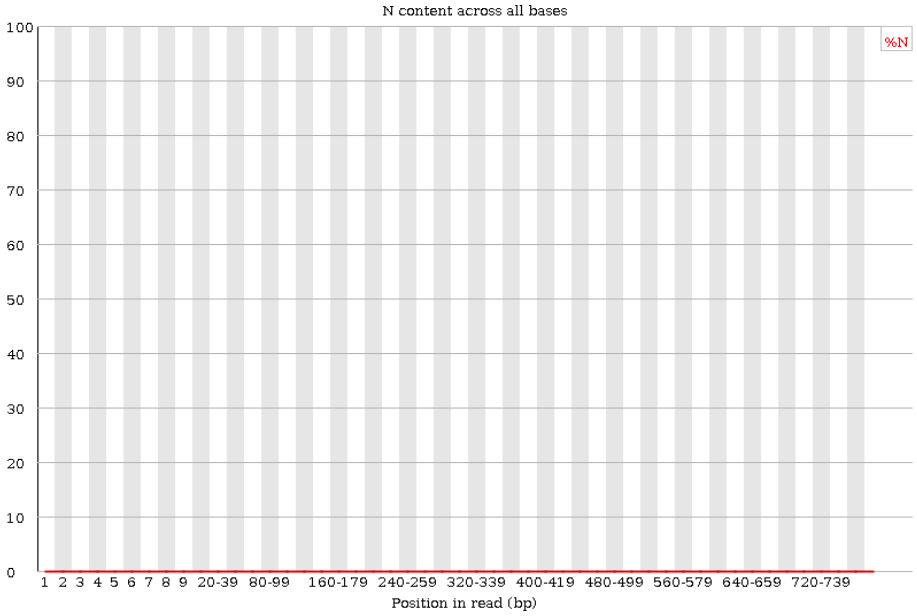


Figure 2.19: The per base N content within the Rooibos natural sample. This figure showed that this sample contained no ambiguous bases.

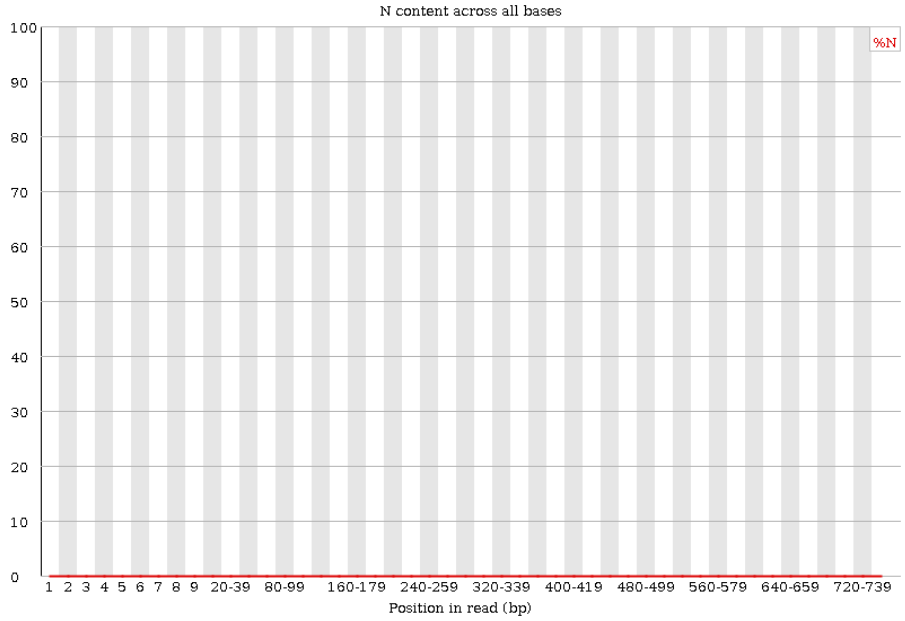


Figure 2.20: The per base N content of the Rooibos commercial sample. This figure showed that there were no ambiguous bases present within this sample.

If the sequences could not make base calls with sufficient confidence, then it was substituted for an N. Figures 2.17-2.20 showed the plots of the percentage of base calls at each position for which an N was called. It was not unusual to see a very low proportion of Ns in the sequences, but it is also not unusual to see a low proportion of Ns near the end of the sequences. But if this proportion of Ns had risen to a few percent, it would have suggested that the analysis pipeline was unable to interpret the data to a well enough degree in which it could make a valid base call.

Per base sequence quality

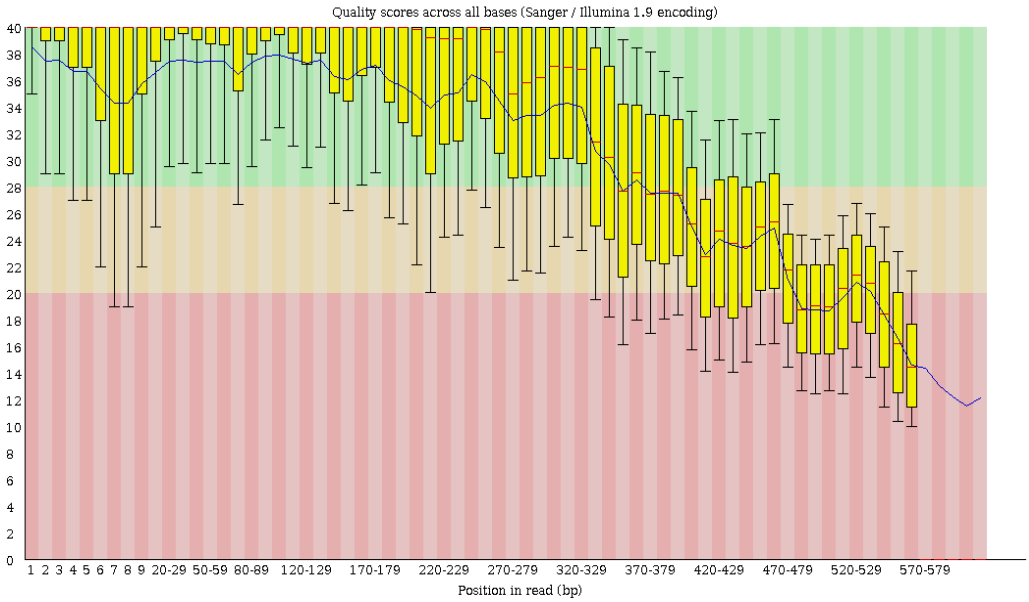


Figure 2.21: The per base sequence quality of the Honeybush natural sample. It was not unusual to see a graph like this because of the primers that were still present within the sample. Green in the figure refers to bases of good quality, orange refers to bases of reasonable quality and red refers to bases of poor quality.

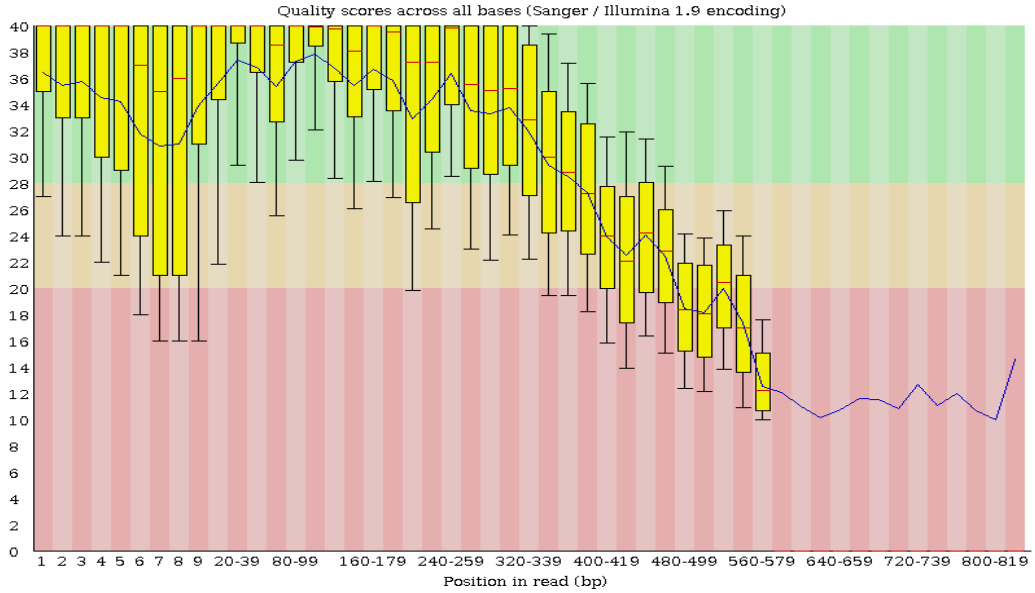


Figure 2.22: The per base sequence quality of the Honeybush commercial sample. It was not unusual to see a graph like this because of the primers that were still present within the sample. Green in the figure refers to bases of good quality, orange refers to bases of reasonable quality and red refers to bases of poor quality.

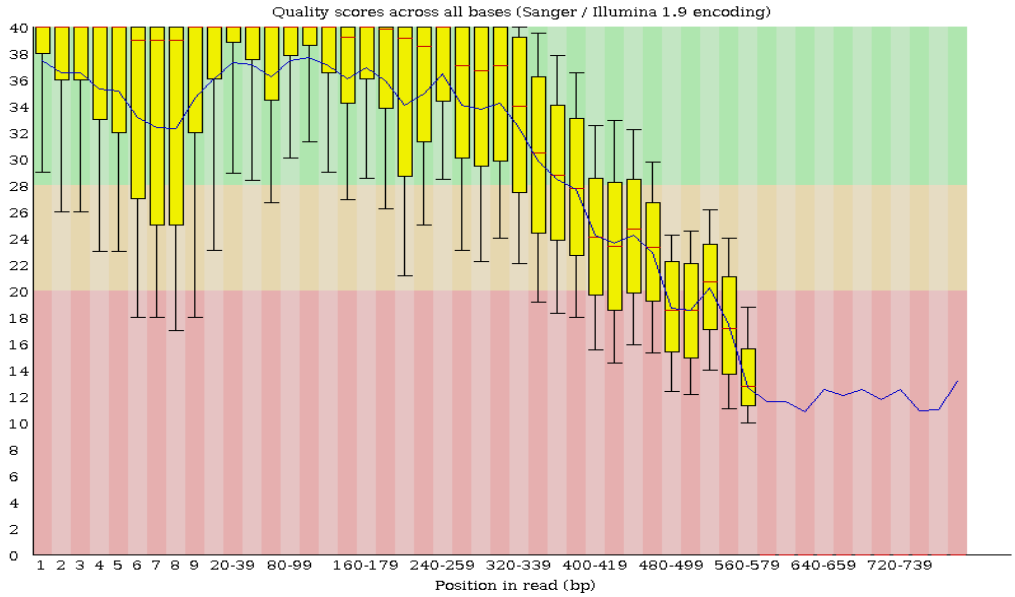


Figure 2.23: The per sequence quality of the Rooibos natural sample. It was not unusual to see a graph like this because of the primers that were still present within the sample. Green in the figure refers to bases of good quality, orange refers to bases of reasonable quality and red refers to bases of poor quality.

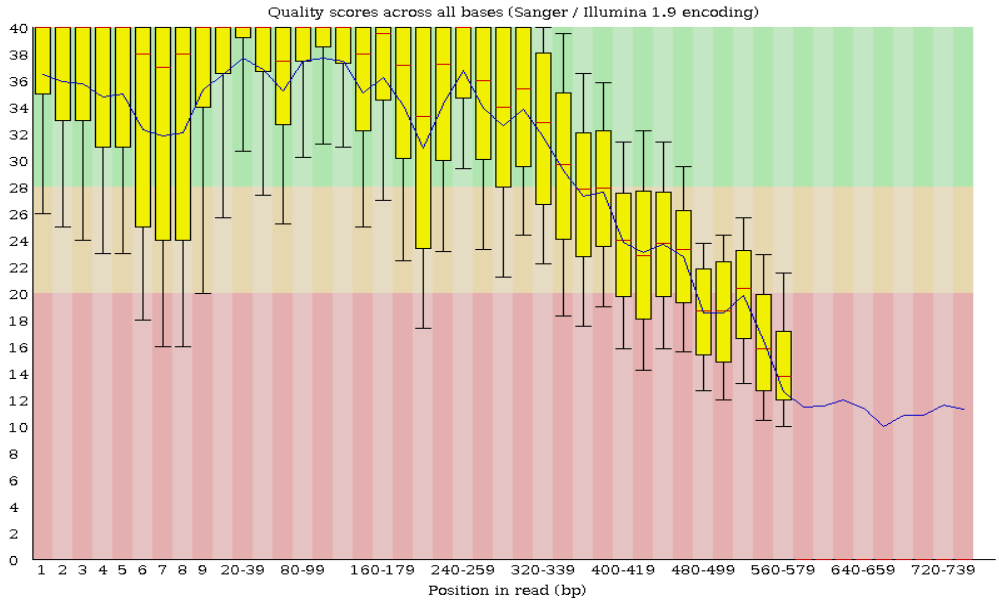


Figure 2.24: The per base sequence quality of the Rooibos commercial sample. It was not unusual to see a graph like this because of the primers that were still present within the sample. Green in the figure refers to bases of good quality, orange refers to bases of reasonable quality and red refers to bases of poor quality.

Figures 2.21-2.24 showed an overview of the range of quality values across all the bases within the sequences in each of the samples. This module looked at each base position within the sequences in the FASTQ file generated from the SFF file and analysed this. For each of the positions in the sequence, a BoxWhisker plot was drawn which showed the following elements: the central red line in the yellow boxes showed the median value, the yellow boxes showed the inter-quartile ranges from 25-75%, the upper and lower whiskers showed the 10% and 90% points in the datasets, and the blue line showed the mean quality value of the sequences. The y-axis on the graphs showed the quality scores. The higher this score was, the better the base call was. The background of the graph was divided into three parts: the green part showed very good quality base calls, the orange part showed reasonable quality base calls, and the red part showed base calls that had a poor quality. It is expected that the quality on most platforms would degrade as the run progresses, so it will be more common to see base calls falling into the orange and red areas towards the end of a read (Caporaso *et al*., 2010).

It was important to note that there were a number of different ways to encode the quality score in the FASTQ file. FASTQC attempts to determine which encoding method was used, but can still get this wrong when the data is universally very good.

Per base sequence content

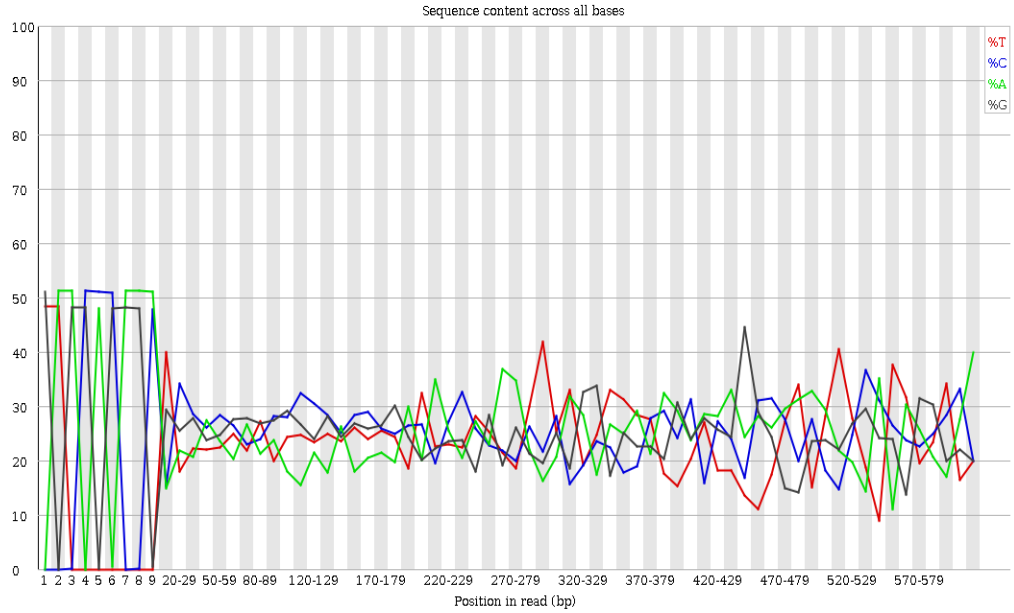


Figure 2.25: The per base sequence content of the Honeybush natural sample. In this figure it was clearly seen that the primers were still attached to the sequences.

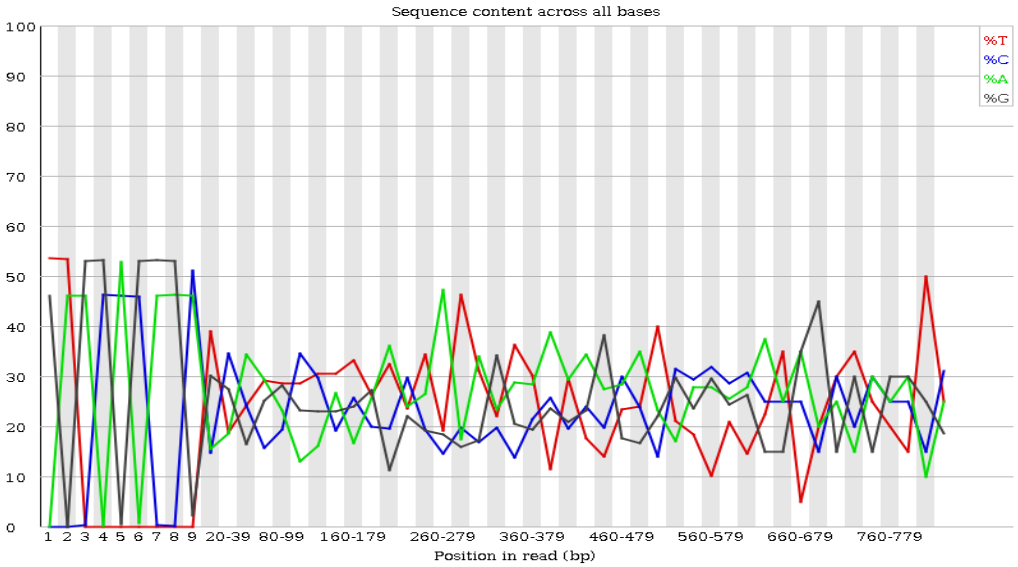


Figure 2.26: The per base sequence content of the Honeybush commercial sample. In this figure it was clearly seen that the primers were still attached to the sequences.

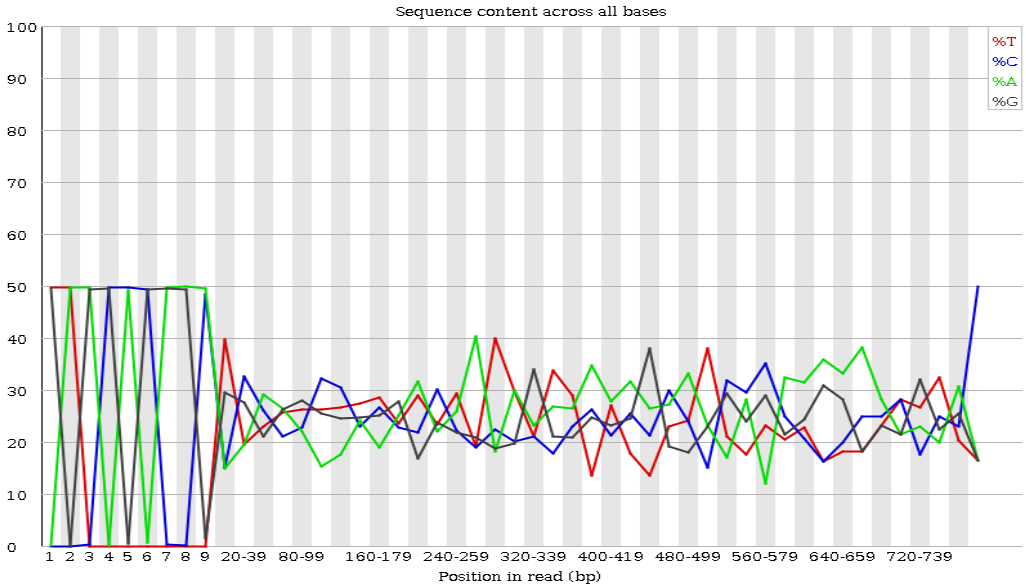


Figure 2.27: The per base sequence content of the Rooibos natural sample. In this figure it was clearly seen that the primers were still attatched to the sequences.

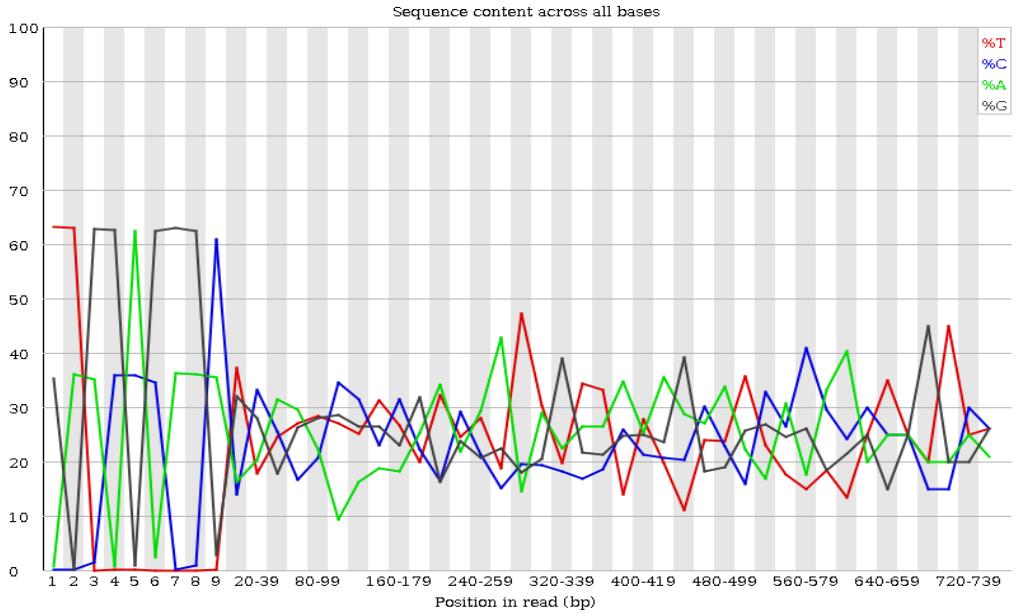


Figure 2.28: The per base sequence content of the Rooibos commercial sample. This figure showed that the primers were still attached to the sequences.

The above figures 2.25-2.28 showed the per base sequence content graphs that were plotted for each of the base positions in the file for which each of the four normal DNA bases were called. The relative amount of each base reflected the overall amount of each base in the genome, but it had to not be imbalanced to a great extent from each other. Some types of samples produced some biases in the sequence composition that were at the start of the reads. Sample sequences produced by priming using hexamers and those which were fragmented using transposases inherited an intrinsic bias in the positions at which the reads started. The biases didn't concern an absolute sequence, but it did provide enrichment of a number of different Kmers at the 5'-

end of the reads. Although this was a true technical bias, it was not something that could be corrected by trimming seeing that it did not necessarily only occur in the primers, and did not seem to affect the downstream analysis. This bias produced a warning message in this module for the user to be able to take note of it (Caporaso *et al*., 2010).

Per sequence GC content

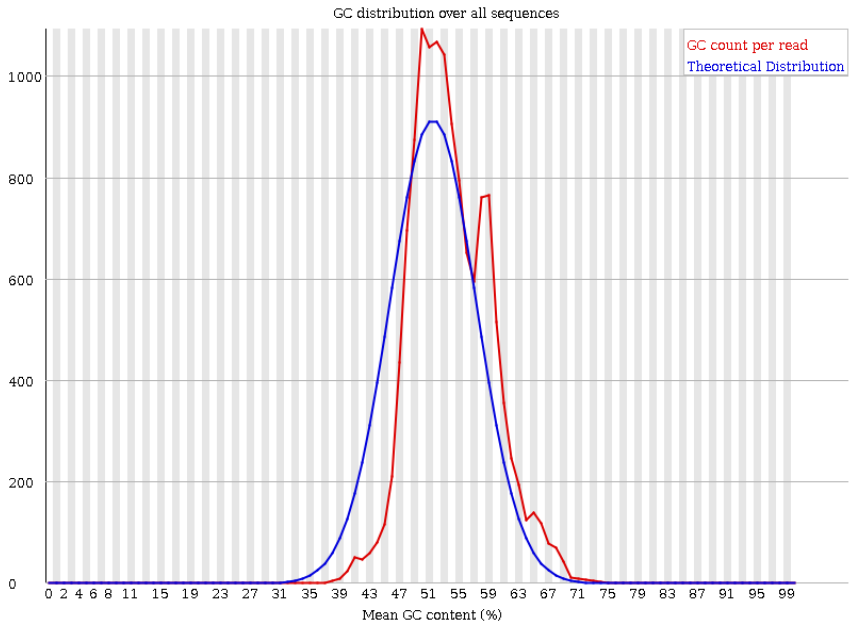


Figure 2.29: The per sequence GC content of the Honeybush natural sample before primer trimming was done. The unusual peak for the GC count per read (red) was a clear indication of gene specific primers that were used to amplify the 18S gene.

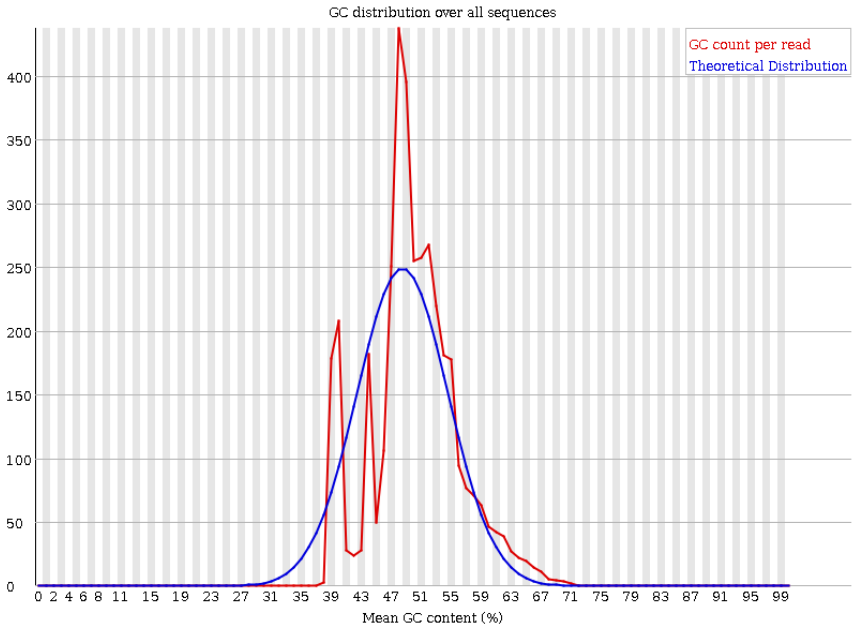


Figure 2.30: The per sequence GC content of the Honeybush commercial sample before primer trimming was done. The uneven GC count per read (red) peak indicated that gene specific primers were used.

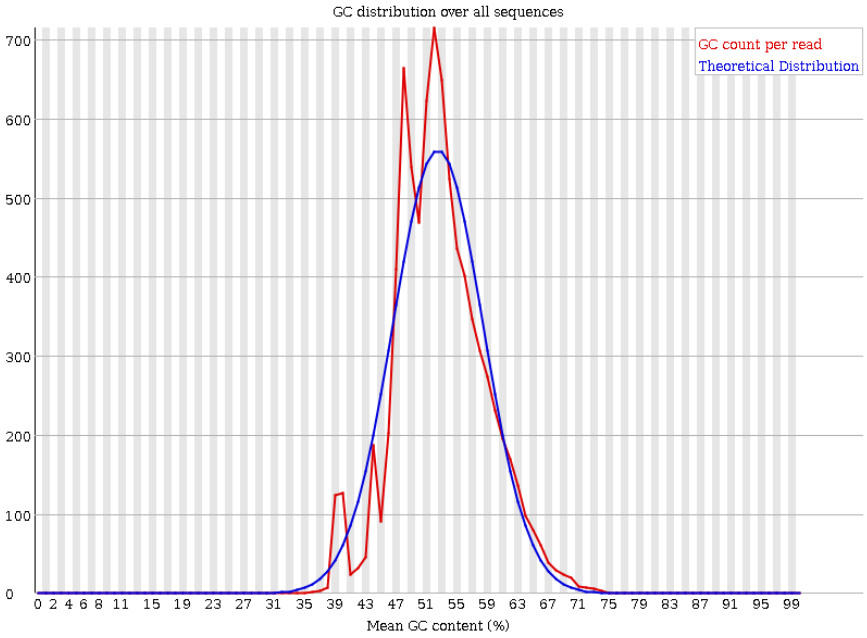


Figure 2.31: The per sequence GC content of the Rooibos natural sample before primer trimming was done. The GC count per read (red) peak indicated that gene specific primers were used.

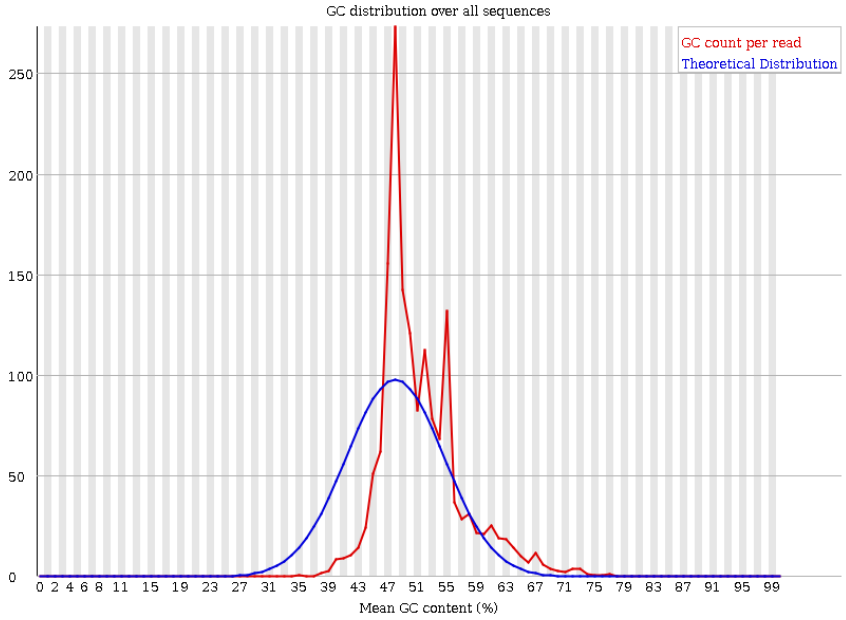


Figure 2.32: The per sequence GC content of the Rooibos commercial sample before primer trimming was done. The red peak indicated that gene specific primers were used during PCR amplification.

The per sequence GC content module measured the GC content across the whole length of each of the sample sequences in each file and compared it to a modelled normal distribution of the GC content. When a normal random sample is studied, it would be expected to see a roughly normal distribution of the GC content where the central peak would correspond to the overall GC content of the underlying genome. But, as in figures 2.29-2.32, because the GC content of the genome was not known, the modal GC content was calculated from the observed data and then used to build a reference distribution. In many other instances an unusually shaped distribution could have indicated a contaminated sample or a biased subset. A normal distribution could be shifted which indicated that some of the systematic bias was independent of base position. It was important to note that if there was a systematic bias which created a shifted normal distribution, it would not be flagged as an error by the module because it did not know what the genome's GC content should have been. In this instance the GC content might have been different from the normal GC content because the gene specific primers were used in the amplification of the 18S gene.

Per sequence quality (Phred) score

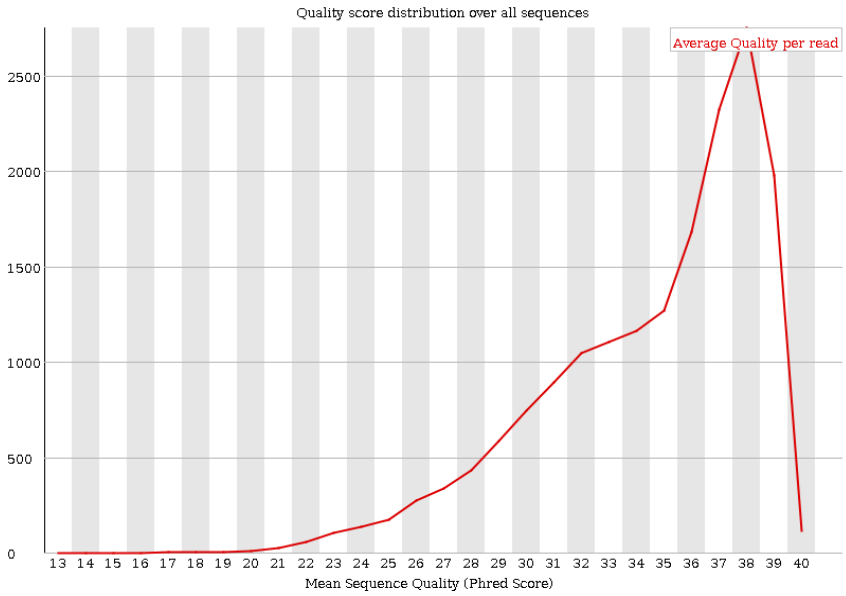


Figure 2.33: The per sequence quality (Phred score) for the quality distribution over all the sequences in the Honeybush natural sample.

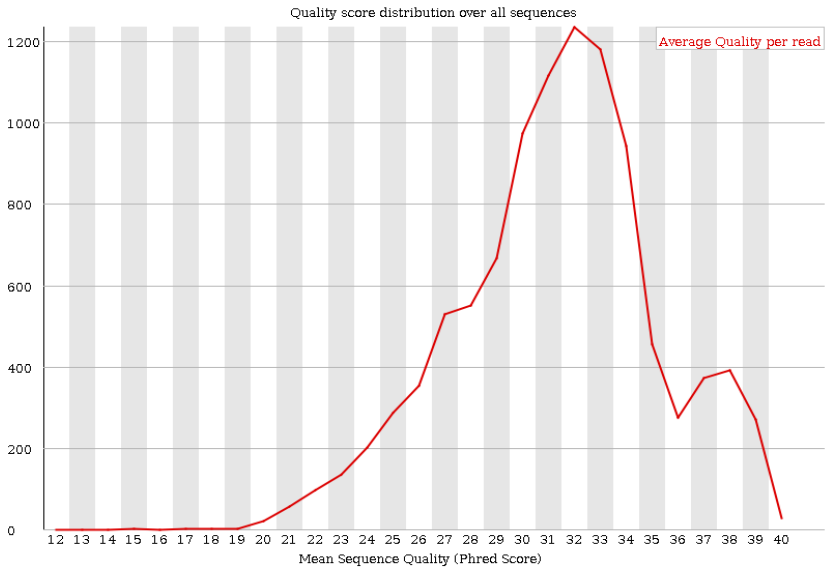


Figure 2.34: The per sequence quality (Phred score) for the quality distribution over all the sequences in the Honeybush commercial sample.

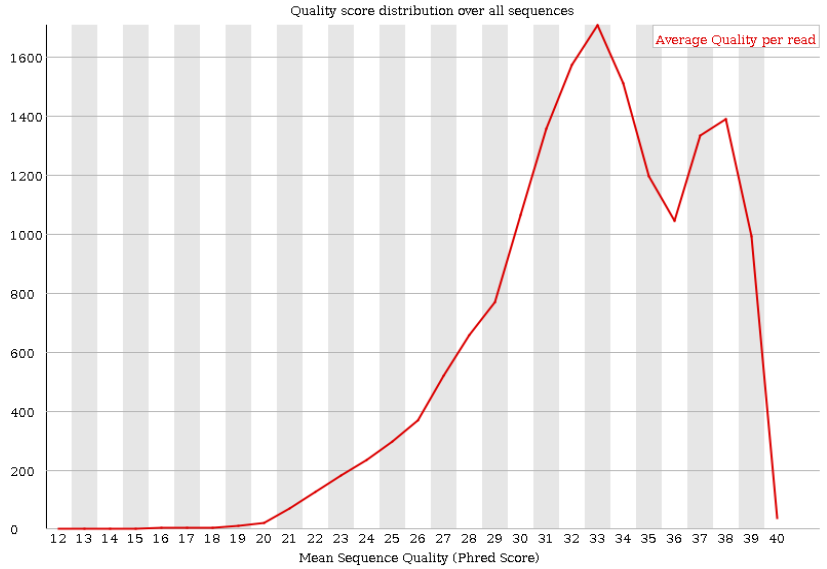


Figure 2.35: The per sequence quality (Phred score) for the quality distribution over all the sequences in the Rooibos natural sample.

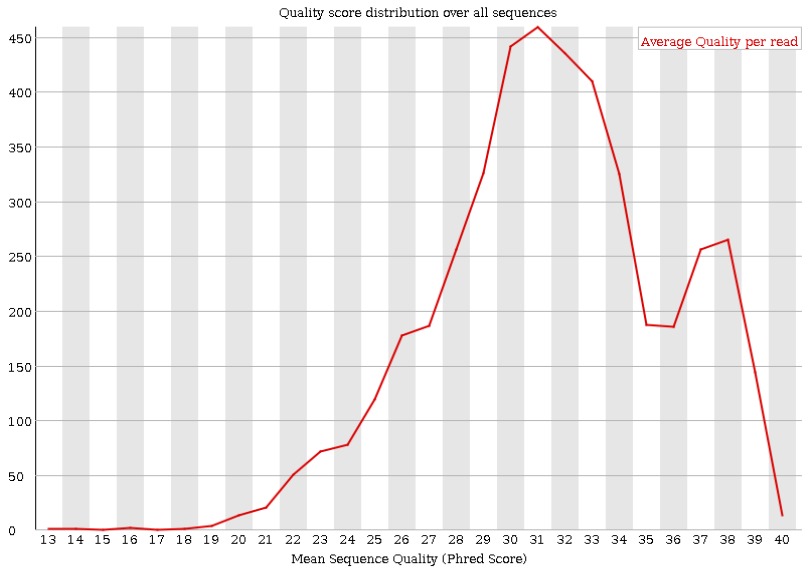


Figure 2.36: The per sequence quality (Phred score) for the quality distribution over all the sequences in the Rooibos commercial sample.

When looking at figure 2.33-2.36 which showed the per sequence quality score report, it was unclear whether a subset of the sequences have universally low quality values or not. This was the case where a subset of sequences had universally poor quality because they were poorly imaged on the edge of the field of view. But these should have represented only a small percentage of the total sequences. If a significant proportion of the sequences in a run had an overall low quality, then it was indicative that some systematics problem could be possible with just part of the run. It is important to note for future studies, that if the input was a BAM/SAM file in which the quality scores have not been recorded, then this module will not show results.

Sequence length distribution

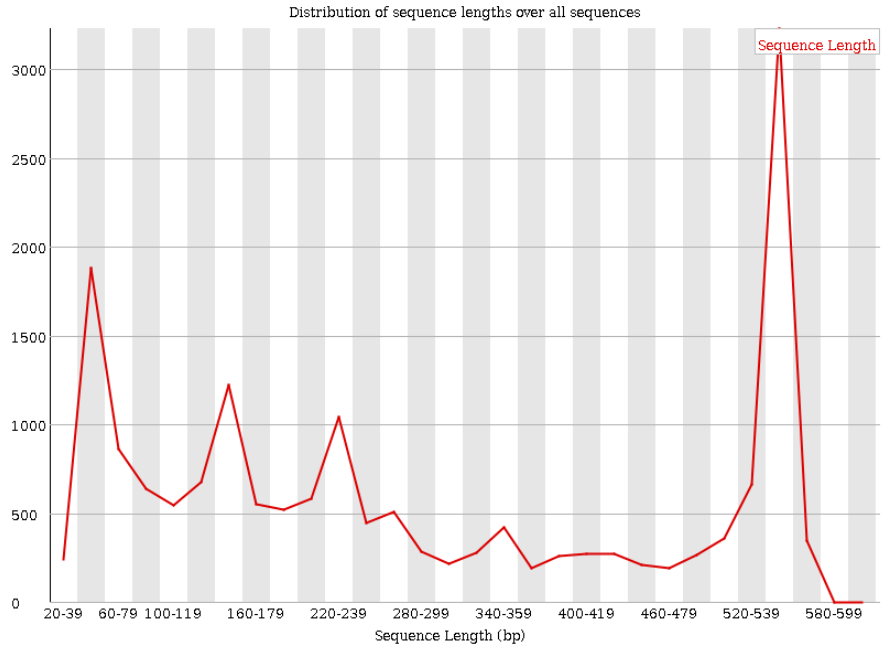


Figure 2.37: Sequence length distribution across all the sequences in the Honeybush natural sample.

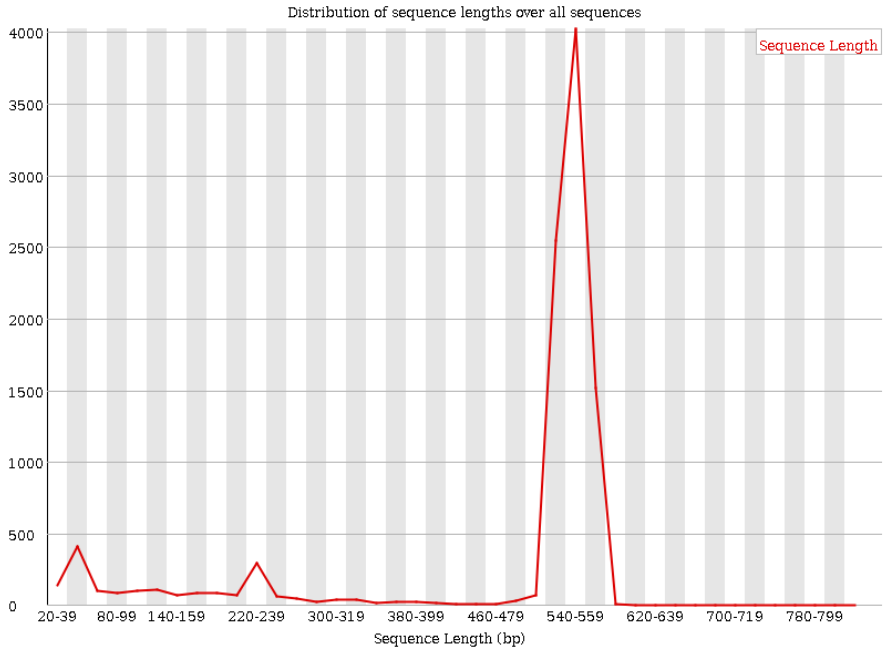


Figure 2.38: Sequence length distribution across all the sequences in the Honeybush commercial sample.

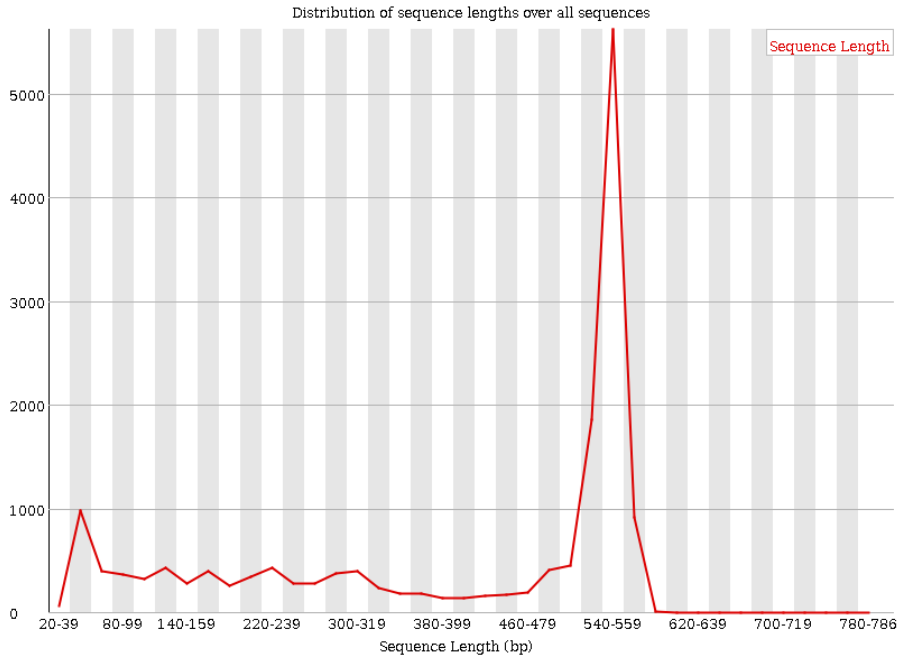


Figure 2.39: Sequence length distribution across all the sequences in the Rooibos natural sample.

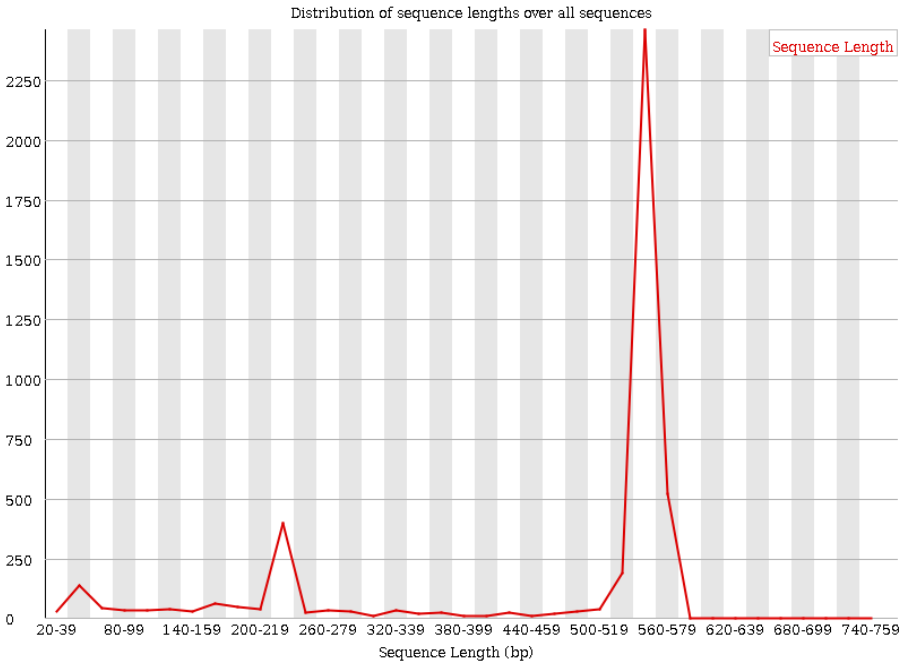


Figure 2.40: Sequence length distribution across all the sequences in the Rooibos commercial sample.

In figures 2.37-2.40 it was seen that some high throughput sequences generated sequence fragments of a uniform length but that some samples contained some reads that had varying lengths. In a uniform length sample, some of the pipelines will trim the sequences to remove poor quality bases from the end of the sequences. The sequence length distribution module generated graphs that showed the distribution of fragment sizes in the files which were analysed

and did, in some cases, produce a simple graph that had a peak at only one size. But for variable FASTQ files, it showed the relative amounts of each different size of sequence fragment.

Table 2.6 indicates the errors or warnings that were found after the FASTQC process was completed. Each individual error or warning had a colour and also an explanation as to what might be the reason for the error or warning to occur.

Table 2.6: Successes, warnings and failures while running FASTQC for each of the different modules for each of the four samples. The MIDs refer to the different sample files where '7' is Honeybush natural, '8' is Honeybush commercial, '9' is Rooibos natural, and '10' is Rooibos commercial.

Module	Samples (MIDs)				Reason for warning/failure
	7	8	9	10	
Basic statistics					-
Per base sequence quality					The lower quartile for any base is less than 5 or if the median for any base is less than 20
Per sequence quality scores					-
Per base sequence content					The difference between A and T, or G and C is greater than 20% in any position.
Per sequence GC content					A warning is raised if the sum of the deviations from the normal distribution represents more than 15% of the reads. This module will indicate a failure if the sum of the deviations from the normal distribution represents more than 30% of the reads.
Per base N content					-
Sequence length distribution					Not all sequences have the same length.
Sequence duplication levels					Non-unique sequences make up more than 50% of the total.
Overrepresented sequences					Any sequence is found to represent more than 1% of the total.
Adapter content					-
Kmer content					This module will issue a warning if any Kmer is imbalanced with a binomial p-value <0.01. This module will issue a warning if any Kmer is imbalanced with a binomial p-value < 10^-5.

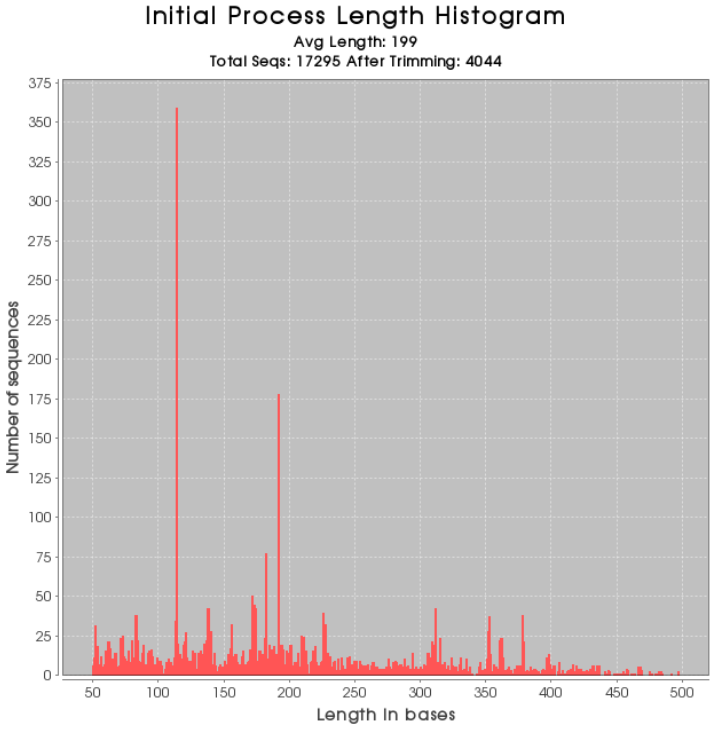
* green = success, yellow = warning, red = error.
** The MID tag number was taken as that used in the lab analysis by the Department of Microbiology and Biochemistry at Rhodes University.

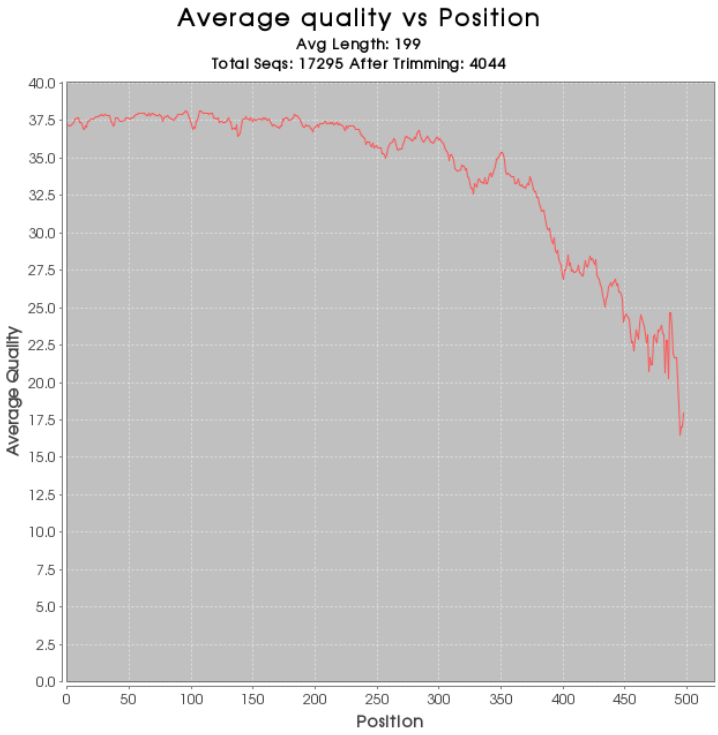
2.4.2) RDP pipeline

The initial processing of the four different sample reads below, were done with the RDP pipeline. Before doing primer trimming, the loss of reads due to quality trimming were much more than when the primers were taken out before using the samples sequences as input. The primers were trimmed from the sample sequences by using Cutadapt (Martin, 2011). The source code for Cutadapt is shown in Appendix VI. The 'Average quality vs Position' graphs also agreed with the 'Per base sequence quality' graphs in section 2.4.1.1.

2.4.2.1) Initial processing results

a.





b.

Figure 2.41: (a and b) Honeybush natural sample QC with initial processing in the RDP pipeline before primer trimming with Cutadapt (Martin, 2011). Figure *a* showed the initial process length histogram that illustrated the length of the sequences within the sample file. Figure *b* showed the average quality vs position graph.

In figure 2.41*a* it showed the initial lengths of the sequences within the sample file. It was important to see how many sequences were lost in this pipeline during initial processing. The RDP pipeline's initial processing included matching of the raw reads to experimental samples. Trimming off the tag (which was not necessary in this case) and primer sequences and then removing the sequences that had a low quality. The fungal ITS gene was chosen in this case (seeing that no other fungal gene was present to choose from) and the orientation of the sequences were checked and reverse complimented where needed. The Honeybush natural sample file started with 17 295 sequences and ended with 4 044 sequences. This was a 76.62% loss of sequences from the original sample file to the file that analysis was continued on in the RDP pipeline. Figure 2.41*b* showed the average quality for each position within the reads of the sample. This is a reasonable graph seeing that the sequence quality will decrease as the base calls become less accurate further down the sequence during the amplification process. From this initial processing, the RDP pipeline reported the average length of the sequences to be 199 bp.

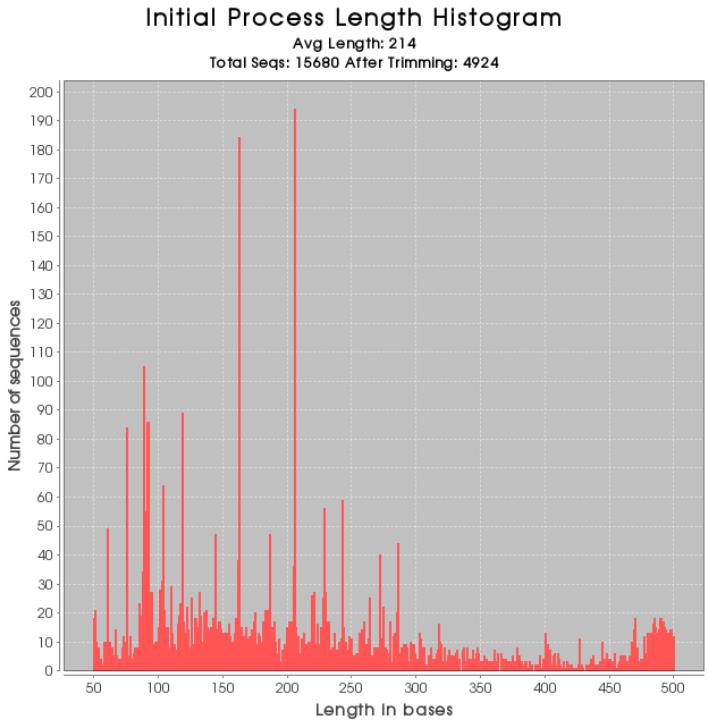
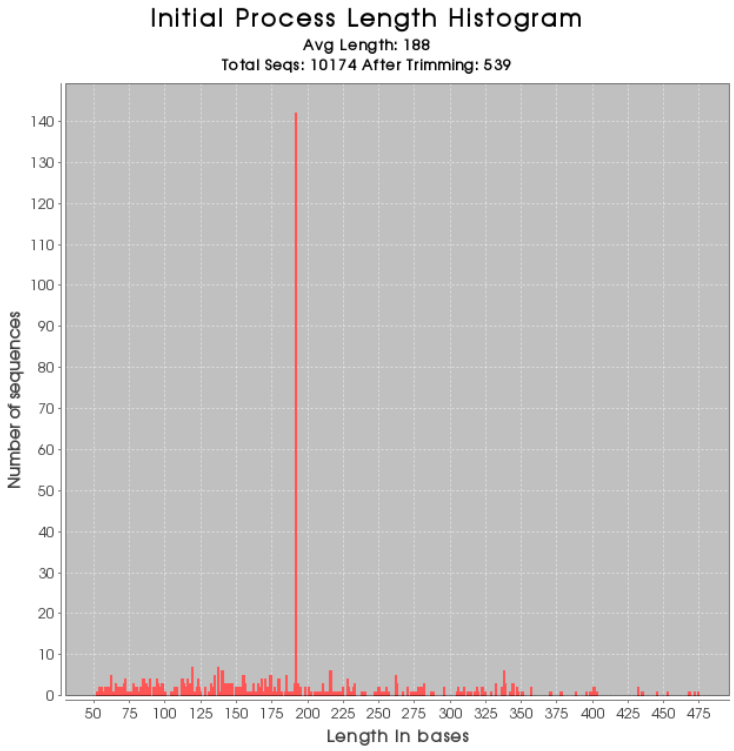
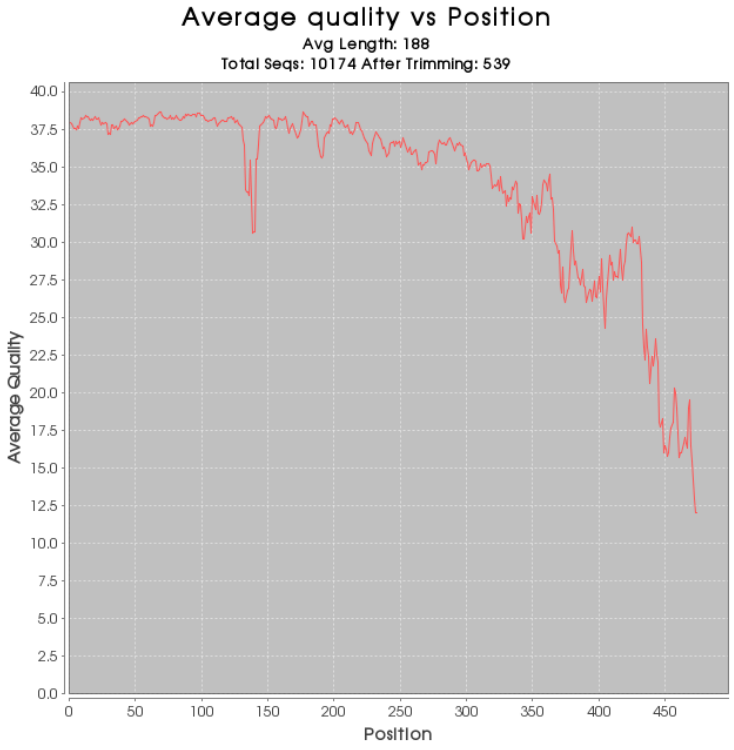


Figure 2.42: Honeybush natural after primer trimming with the RDP initial processing step. This figure illustrated the initial process length histogram of the sequences that were present in the Honeybush natural sample after primer trimming was done.

In figure 2.42 it was seen that the average length of the sequences in the sample file was reported to be 214 bp, which was longer than that reported for the sample file where the primers were not previously removed. Also, the amount of starting sequences were less, which suggested that about ~ 2 000 sequences were made up of primers. After trimming and initial processing the total number of sequences were 4 924, which were more than what was previously reported where the primers were not removed. This led to a 68.6% loss of sequences during the initial processing stage.



a.



b.

Figure 2.43: (a and b) Honeybush commercial sample initial processing results before primer trimming with Cutadapt. Figure *a* showed the initial process length histogram for this sample, and figure *b* showed the average quality vs position for each base in the sequences in the sample file.

Figure 2.43*a* showed the initial process lengths for each sequence in the Honeybush commercial sample. This showed that the average length of the sequences were 188 bp, the starting amount of sequences in the sample file was 10 174 and after trimming there were 539

sequences left in the sample file. This was a massive loss of 94.7% of the sequences during this step of analysis. This was regarded as being due to the fungal ITS gene being chosen as a reference gene and not the 18S AM fungal gene. Figure 2.43*b* showed the average quality per base of the sequences in the sample file. As expected the quality of the base calling does decrease as the sequences grow longer during the PCR amplification.

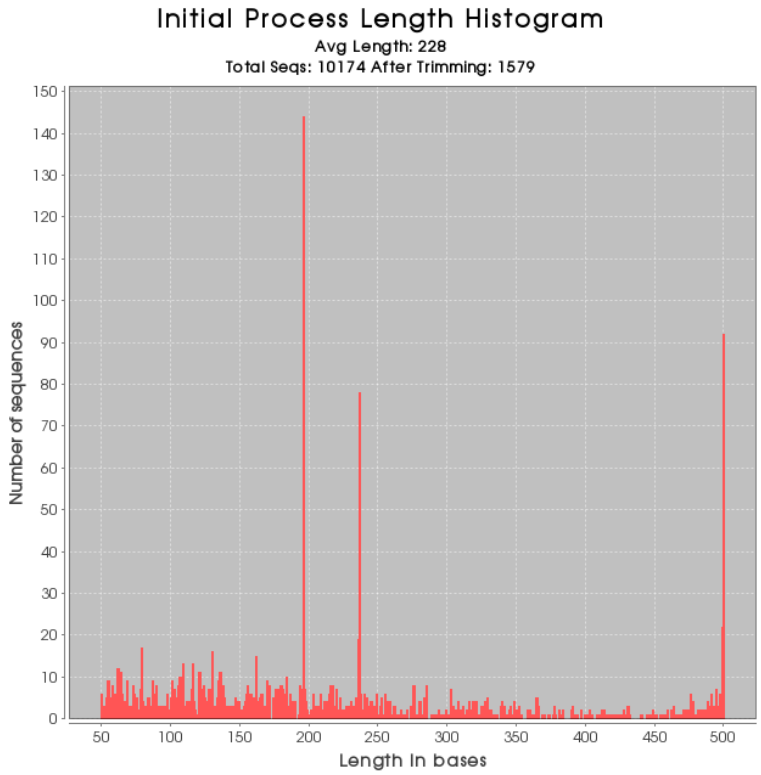
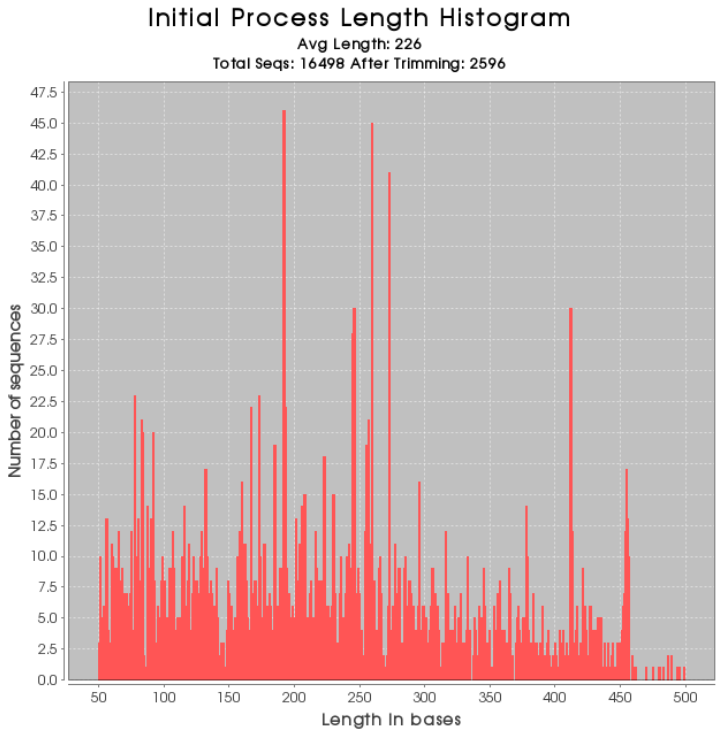
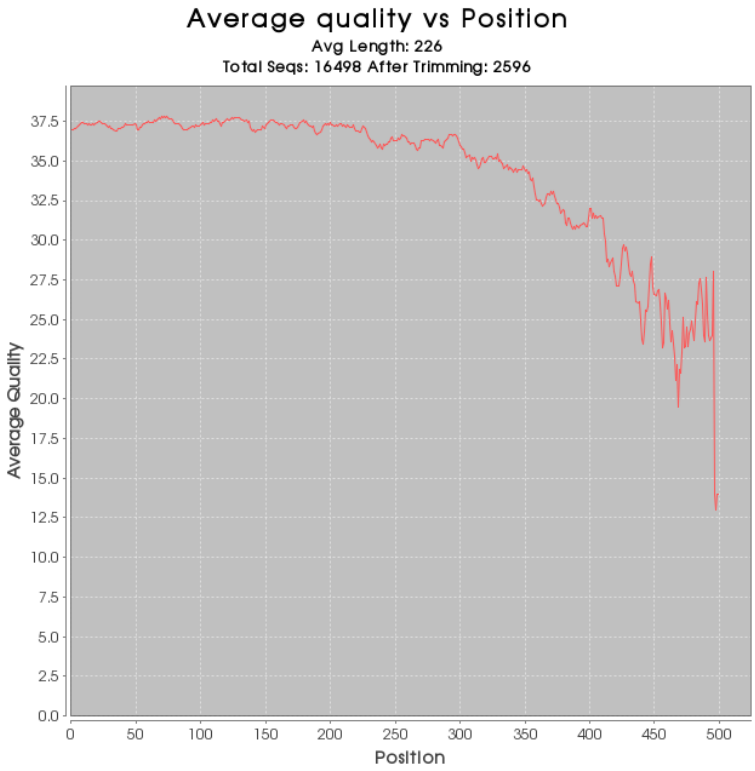


Figure 2.44: Honeybush commercial sample after primer trimming. This figure showed the initial process length histogram of the sample sequences.

In figure 2.44 it was seen that the average length of the sequences after primer trimming, with Cutadapt (Martin, 2011), was calculated to be 228 bp. The total sequences in the sample file before intial processing was 10 174 and after the initial processing, the number of sequences in the sample were 1 579. After primers were manually trimmed from the sample sequences, the total loss of sequences were calculated to be 84.48%, which were less than that of the Honeybush commercial sample before the primers were trimmed.



a.



b.

Figure 2.45: (a and b) Rooibos natural sample before primer trimming. Figure *a* showed the initial process length histogram for all the sequences in the sample file before primers were trimmed. Figure *b* showed the average quality vs position for each base of the sequences in the sample file.

Figure 2.45*a* showed the initial process length for the Rooibos natural sample before primer trimming was done (with a Cutadapt). The average sequence length was found to be 226 bp in

the sample file. The total number of sequences before initial processing was counted as 16 498 and after initial processing there were 2 596 sequences left in the sample file. There was thus a 84.26% loss of sequences during initial processing. Figure 2.45*b* showed the average quality per base position for each base call done during amplification. Again this graph made sense in that the base call quality does decrease as the length of the sequences increased during sequencing. But what was interesting with this figure was that the base call quality did fall drastically at position ~ 500, which did not occur with any of the other samples.

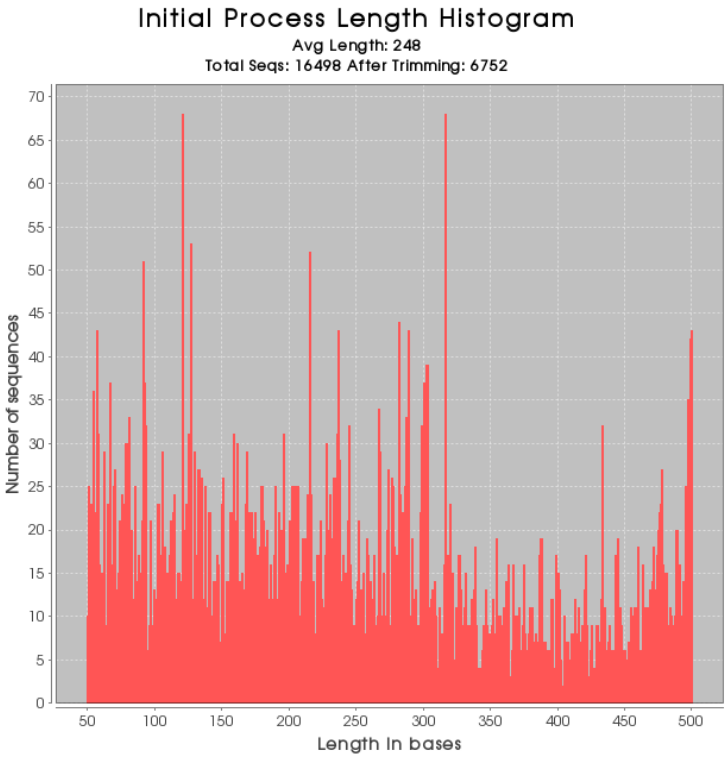
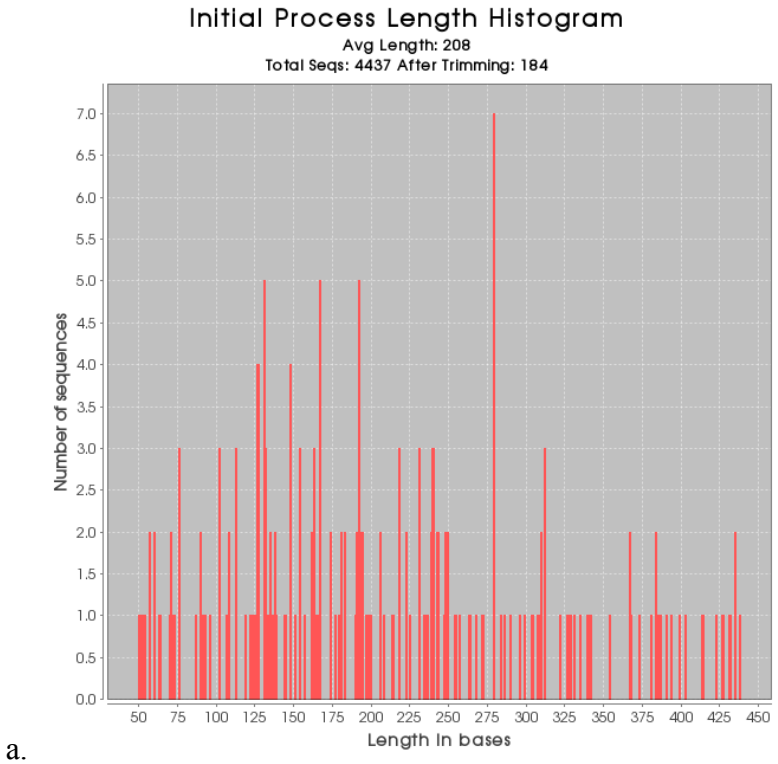
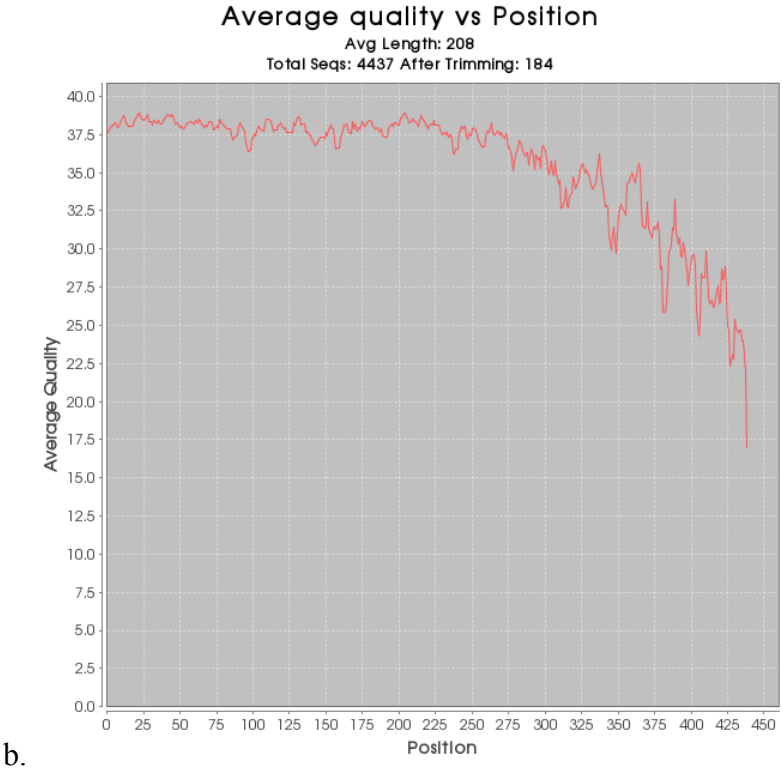


Figure 2.46: Rooibos natural sample after primer trimming was done. This figure showed the initial process length histogram of the reads in the sample file.

In figure 2.46 it was seen that the average length calculated for the sequences in the sample file after primer trimming was done with Cutadapt, and was 248 bp. This was again longer than that for the Rooibos natural sample before primer trimming was done. The total sequences before initial processing was 16 498 and after processing it was 6 752. The amount of sequences after trimming was again more than after initial processing, before primer trimming was done. This figure showed that there was a 59.08% loss of sequences which was ~ 30% less loss of sequences than with the other samples.



a.



b.

Figure 2.47: (a and b) Rooibos commercial before primer trimming was done. Figure *a* refers to the initial process length histogram that was drawn by the RDP pipeline after initial processing was done, before primer trimming. Figure *b* showed the average quality vs position for each base call for this sample file.

Figure 2.47*a* showed the initial process length of the sequences before primer trimming was conducted on the sample file. The RDP pipeline calculated that the average length of the sequences in the sample file were 208 bp. The initial amount of sequences in the sample file before initial processing was 4 437 sequences. The amount of sequences present in the sample file after initial processing was 184. This is a total loss of 95.85% of sequences during initial processing.

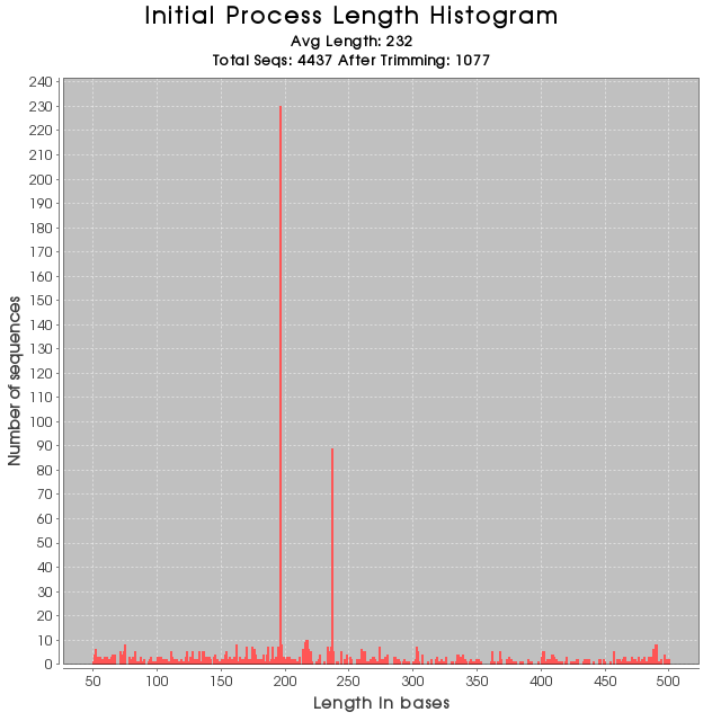


Figure 2.48: Rooibos commercial sample after primer trimming. This figure showed the initial process length histogram of the sequences within the sample file.

Figure 2.48 showed that the average length of the sequences within the Rooibos commercial file after primer trimming was 232 bp. The sequences in this file thus had a longer average length than the sequences in the Rooibos commercial sample before manual primer trimming was done. The initial amount of sequences in the sample file before initial processing was done, was 4 437, and after processing there were 1 077 sequences left in the sample file. This led to a 75.73% calculated loss of sequences during initial processing. This again was a smaller loss of sequences than there was in the sample file before primer trimming was done.

After considering all the above results, it was suggested to do further downstream studies with the sample files in which the primer sequences were trimmed from the sequences beforehand.

This led to larger sample files to work with than after the QC results in the RDP pipeline in each different sample and thus a larger dataset to study.

2.4.3) Mothur

The results that follow were grouped into the different samples before and after QC was done with Mothur. The results found here were calculated using the **summary.seqs** command before and after QC was done.

Table 2.6: Summary of the Honeybush natural sample sequences. This table represented the results before QC and primer trimming were done.

MID7 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	34	34	0	2	1
2.5%-tile:	1	40	40	0	3	433
25%-tile:	1	123	123	0	3	4324
Median:	1	237	237	0	4	8648
75%-tile:	1	516	516	0	6	12972
97.5%-tile:	1	559	559	0	7	16863
Maximum:	1	614	614	6	11	17295
Mean:	1	289.003	289.003	0.0141081	4.43481	
# of Seqs:	17295					

The ‘NBases’ referred to the amount of Ns counted within the sequences, whereas the ‘Ambigs’ referred to the ambiguous bases found within the sequences. ‘NumSeqs’ referred to the number of sequences that had specific statistical properties.

Table 2.7: Summary of the Honeybush natural sample sequences. This table represented the results after QC and primer trimming were done.

MID7 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	-1	-1	0	0	1	32
25%-tile:	1	53	53	0	3	317
Median:	1	114	114	0	4	633
75%-tile:	1	231	231	0	4	949
97.5%-tile:	1	380	380	0	7	1233
Maximum:	1	451	451	1	9	1264
Mean:	0.908228	151.927	151.973	0.000791139	3.66614	
# of Seqs:	1264					

From tables 2.6 and 2.7 it could be seen that there were 17 295 sequences in the sample before QC. The shortest read was 34 bp and the longest was 614 bp. The median length of this sample was 237 bp.

Table 2.8: Summary of the Honeybush commercial sample sequences. This table showed the statistical results before QC and primer trimming were done.

MID8 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	36	36	0	3	1
2.5%-tile:	1	42	42	0	3	255
25%-tile:	1	522	522	0	5	2544
Median:	1	547	547	0	6	5088
75%-tile:	1	555	555	0	6	7631
97.5%-tile:	1	565	565	0	6	9920
Maximum:	1	835	835	4	8	10174
Mean:	1	472.099	472.099	0.0155298	5.34667	
# of Seqs:	10174					

Table 2.9: Summary of the Honeybush commercial sample sequences. This table showed the statistical results after QC and primer trimming were done.

MID8 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	-1	-1	0	0	1	5
25%-tile:	1	18	18	0	2	41
Median:	1	94	94	0	3	82
75%-tile:	1	198	198	0	4	123
97.5%-tile:	1	347	347	0	6	159
Maximum:	1	516	516	0	6	163
Mean:	0.840491	121.638	121.718	0	3.31288	
# of Seqs:	163					

From tables 2.8 and 2.9 it could be seen that there were 10 174 sequences in the sample before QC. The shortest read was 36 bp and the longest was 835 bp. The median length of this sample was 547 bp.

Table 2.10: Summary of the Rooibos natural sample sequences. This table showed the statistical results of the sample before QC and primer trimming were done.

MID9 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	28	28	0	3	1
2.5%-tile:	1	43	43	0	3	413
25%-tile:	1	232	232	0	4	4125
Median:	1	522	522	0	5	8250
75%-tile:	1	551	551	0	6	12374
97.5%-tile:	1	562	562	0	6	16086
Maximum:	1	785	785	11	9	16498
Mean:	1	398.305	398.305	0.0153352	4.99345	
# of Seqs:	16498					

Table 2.11: Summary of the Rooibos natural sample sequences. This table showed the statistical results after QC and primer trimming were done.

MID9 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	-1	-1	0	0	1	16
25%-tile:	1	75	75	0	3	152
Median:	1	193	193	0	4	304
75%-tile:	1	264	264	0	5	455
97.5%-tile:	1	385	385	0	6	591
Maximum:	1	456	456	0	7	606
Mean:	0.927393	182.573	182.609	0	3.85314	
# of Seqs:	606					

From tables 2.10 and 2.11 it could be seen that there were 16 498 sequences in the sample before QC. The shortest read was 28 bp and the longest was 785 bp. The median length of this sample was 522 bp.

Table 2.12: Summary of the Rooibos commercial sample. This table showed the statistical results of the sequences before QC and primer trimming were done.

MID10 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	33	33	0	3	1
2.5%-tile:	1	43	43	0	3	111
25%-tile:	1	406	406	0	5	1110
Median:	1	549	549	0	6	2219
75%-tile:	1	554	554	0	6	3328
97.5%-tile:	1	564	564	0	7	4327
Maximum:	1	758	758	3	8	4437
Mean:	1	459.287	459.287	0.0178048	5.43385	
# of Seqs:	4437					

Table 2.13: Summary of the Rooibos commercial sample sequences. This table showed the statistical results of the sequences after QC and primer trimming were done.

MID10 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	-1	-1	0	0	1	2
25%-tile:	1	46	46	0	3	12
Median:	1	138	138	0	3	23
75%-tile:	1	218	218	0	4	34
97.5%-tile:	1	403	403	0	5	43
Maximum:	1	423	423	0	5	44
Mean:	0.863636	141.909	141.977	0	3.38636	
# of Seqs:	44					

From tables 2.12 and 2.13 it could be seen that there were 4 437 sequences in the sample before QC. The shortest read was 33 bp and the longest was 758 bp. The median length of this sample was 549 bp.

2.4.4) Comparisons of QC results from different pipelines

In this section, the results from the three different pipelines' QC steps were compared. All the results shown here were calculated using Mothur and the command **summary.seqs**. Mothur and summary.seqs were used because it gave a quick statistical report on the input and output from the different pipelines' initial processing step. The input for the summary.seqs command were the FASTA files used for the input and found as output, respectively, from the various pipelines' first step.

2.4.4.1) QIIME

Table 2.14: Summary of the Honeybush natural sample. The statistical results before QIIME FastQC and primer trimming were done, for comparisons between the different samples and different pipelines.

MID7 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	34	34	0	2	1
2.5%-tile:	1	40	40	0	3	433
25%-tile:	1	123	123	0	3	4324
Median:	1	237	237	0	4	8648
75%-tile:	1	516	516	0	6	12972
97.5%-tile:	1	559	559	0	7	16863
Maximum:	1	614	614	6	11	17295
Mean:	1	289.003	289.003	0.0141081	4.43481	
# of Seqs:	17295					

Table 2.15: Summary of the Honeybush natural sample. The statistical results after QIIME FastQC and primer trimming were done, for comparisons between the different samples and different pipelines.

MID7 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	-1	-1	0	0	1	32
25%-tile:	1	53	53	0	3	317
Median:	1	114	114	0	4	633
75%-tile:	1	231	231	0	4	949
97.5%-tile:	1	380	380	0	7	1233
Maximum:	1	451	451	1	9	1264
Mean:	0.908228	151.927	151.973	0.000791139	3.66614	
# of Seqs:	1264					

In table 2.14 and 2.15, it was seen that there were 17 295 sequences in the original sample file but after QC with QIIME, there were only 1 264 sequences left. QIIME created a lot of empty sequences (0 bp) with QC and these had to be removed before doing further analysis (Chapter 3). The median length of sequences before QC was 237 bp and after QC was 114 bp. The longest sequences before QC were 614 bp long and after QC were 451 bp.

Table 2.16: Summary of the Honeybush commercial sample. Before QIIME FastQC and primer trimming were done, for comparisons between the different samples and different pipelines.

MID8 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	36	36	0	3	1
2.5%-tile:	1	42	42	0	3	255
25%-tile:	1	522	522	0	5	2544
Median:	1	547	547	0	6	5088
75%-tile:	1	555	555	0	6	7631
97.5%-tile:	1	565	565	0	6	9920
Maximum:	1	835	835	4	8	10174
Mean:	1	472.099	472.099	0.0155298	5.34667	
# of Seqs:	10174					

Table 2.17: Summary of the Honeybush commercial sample. After QIIME FastQC and primer trimming were done, for comparisons between the different samples and different pipelines.

MID8 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	2	2	0	1	1
2.5%-tile:	1	3	3	0	1	4
25%-tile:	1	28	28	0	3	37
Median:	1	114	114	0	4	74
75%-tile:	1	216	216	0	4	111
97.5%-tile:	1	350	350	0	6	144
Maximum:	1	516	516	0	6	147
Mean:	1	134.939	134.939	0	3.56463	
# of Seqs:	147					

In table 2.16 and 2.17, it was seen that there were 10 174 sequences in the original sample file but after QC with QIIME, there were 147 sequences left. QIIME created no empty sequences (0 bp) with QC but some sequences had to be removed before doing further analysis due to being shorter than 10 bp. The median length of sequences before QC was 547 bp and after QC was 114 bp. The longest sequences before QC were 835 bp long and after QC were 516 bp.

Table 2.18: Summary of the Rooibos natural sample. Before QIIME FastQC and primer trimming were done, for comparisons between the different samples and different pipelines.

MID9 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	28	28	0	3	1
2.5%-tile:	1	43	43	0	3	413
25%-tile:	1	232	232	0	4	4125
Median:	1	522	522	0	5	8250
75%-tile:	1	551	551	0	6	12374
97.5%-tile:	1	562	562	0	6	16086
Maximum:	1	785	785	11	9	16498
Mean:	1	398.305	398.305	0.0153352	4.99345	
# of Seqs:	16498					

Table 2.19: Summary of the Rooibos natural sample. After QIIME FastQC and primer trimming were done, for comparisons between the different samples and different pipelines.

MID9 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	1	8	8	0	2	15
25%-tile:	1	88	88	0	3	147
Median:	1	196	196	0	4	293
75%-tile:	1	272	272	0	5	439
97.5%-tile:	1	389	389	0	6	571
Maximum:	1	456	456	0	7	585
Mean:	0.996581	189.162	189.164	0	3.95556	
# of Seqs:	585					

In table 2.18 and 2.19, it was seen that there were 16 498 sequences in the original sample file but after QC with QIIME, there were 585 sequences left. QIIME created a lot of empty sequences (0 bp) with QC and these had to be removed before doing further analysis. The median length of sequences before QC was 522 bp and after QC was 196 bp. The longest sequences before QC were 785 bp long and after QC were 456 bp.

Table 2.20: Summary of the Rooibos commercial sample. Before QIIME FastQC and primer trimming were done, for comparisons between the different samples and different pipelines.

MID10 before	Start	End	NBases	Ambigs	Polymer	NumSeqs

Minimum:	1	33	33	0	3	1
2.5%-tile:	1	43	43	0	3	111
25%-tile:	1	406	406	0	5	1110
Median:	1	549	549	0	6	2219
75%-tile:	1	554	554	0	6	3328
97.5%-tile:	1	564	564	0	7	4327
Maximum:	1	758	758	3	8	4437
Mean:	1	459.287	459.287	0.0178048	5.43385	
# of Seqs:	4437					

Table 2.21: Summary of the Rooibos commercial sample. After QIIME FastQC and primer trimming were done, for comparisons between the different samples and different pipelines.

MID10 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	10	10	0	2	1
2.5%-tile:	1	24	24	0	2	2
25%-tile:	1	57	57	0	3	11
Median:	1	144	144	0	3	21
75%-tile:	1	218	218	0	4	31
97.5%-tile:	1	403	403	0	5	40
Maximum:	1	423	423	0	5	41
Mean:	1	152.366	152.366	0	3.56098	
# of Seqs:	41					

In table 2.20 and 2.21, it was seen that there were 4 437 sequences in the original sample file but after QC with QIIME, there were only 41 sequences left. QIIME created no empty sequences (0 bp) with QC and no sequences had to be removed when doing further analysis because the sequences had a minimum length of 10 bp. The median length of sequences before QC was 549 bp and after QC was 144 bp. The longest sequences before QC were 758 bp long and after QC were 423 bp.

2.4.4.2) Mothur

Table 2.22: Summary of the Honeybush natural sample sequences. Before QC and primer trimming were done by Mothur so comparisons could be made between the different samples and pipelines.

MID7 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	34	34	0	2	1
2.5%-tile:	1	40	40	0	3	433

25%-tile:	1	123	123	0	3	4324
Median:	1	237	237	0	4	8648
75%-tile:	1	516	516	0	6	12972
97.5%-tile:	1	559	559	0	7	16863
Maximum:	1	614	614	6	11	17295
Mean:	1	289.003	289.003	0.0141081	4.43481	
# of Seqs:	17295					

Table 2.23: Summary of the Honeybush natural sample sequences. After QC and primer trimming were done by Mothur so comparisons could be made between the different samples and pipelines.

MID7 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	-1	-1	0	0	1	32
25%-tile:	1	53	53	0	3	317
Median:	1	114	114	0	4	633
75%-tile:	1	231	231	0	4	949
97.5%-tile:	1	380	380	0	7	1233
Maximum:	1	451	451	1	9	1264
Mean:	0.908228	151.927	151.973	0.000791139	3.66614	
# of Seqs:	1264					

In table 2.22 and 2.23, it was seen that there were 17 295 sequences in the original sample file but after QC with Mothur, there were 1 264 sequences left. Mothur created a lot of empty sequences (0 bp) with QC and these had to be removed before doing further analysis. The median length of sequences before QC was 237 bp and after QC was 114 bp. The longest sequences before QC were 614 bp long and after QC were 451 bp.

Table 2.24: Summary of the Honeybush commercial sample sequences. Before QC and primer trimming were done by Mothur so comparisons could be made between the different samples and pipelines.

MID8 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	36	36	0	3	1
2.5%-tile:	1	42	42	0	3	255
25%-tile:	1	522	522	0	5	2544
Median:	1	547	547	0	6	5088
75%-tile:	1	555	555	0	6	7631

97.5%-tile:	1	565	565	0	6	9920
Maximum:	1	835	835	4	8	10174
Mean:	1	472.099	472.099	0.0155298	5.34667	
# of Seqs:	10174					

Table 2.25: Summary of the Honeybush commercial sample sequences. After QC and primer trimming were done by Mothur so comparisons could be made between the different samples and pipelines.

MID8 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	-1	-1	0	0	1	5
25%-tile:	1	18	18	0	2	41
Median:	1	94	94	0	3	82
75%-tile:	1	198	198	0	4	123
97.5%-tile:	1	347	347	0	6	159
Maximum:	1	516	516	0	6	163
Mean:	0.840491	121.638	121.718	0	3.31288	
# of Seqs:	163					

In table 2.24 and 2.25, it was seen that there were 10 174 sequences in the original sample file but after QC with Mothur, there were 163 sequences left. Mothur created empty sequences (0 bp) with QC but these had to be removed when doing further analysis due to being shorter than 10 bp. The median length of sequences before QC was 547 bp and after QC was 94 bp. The longest sequences before QC were 835 bp long and after QC were 516 bp.

Table 2.26: Summary of the Rooibos natural sample sequences. Before QC and primer trimming were done by Mothur so comparisons could be made between the different samples and pipelines.

MID9 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	28	28	0	3	1
2.5%-tile:	1	43	43	0	3	413
25%-tile:	1	232	232	0	4	4125
Median:	1	522	522	0	5	8250
75%-tile:	1	551	551	0	6	12374
97.5%-tile:	1	562	562	0	6	16086
Maximum:	1	785	785	11	9	16498

Mean:	1	398.305	398.305	0.0153352	4.99345	
# of Seqs:	16498					

Table 2.27: Summary of the Rooibos natural sample sequences. After QC and primer trimming were done by Mothur so comparisons could be made between the different samples and pipelines.

MID9 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	-1	-1	0	0	1	16
25%-tile:	1	75	75	0	3	152
Median:	1	193	193	0	4	304
75%-tile:	1	264	264	0	5	455
97.5%-tile:	1	385	385	0	6	591
Maximum:	1	456	456	0	7	606
Mean:	0.927393	182.573	182.609	0	3.85314	
# of Seqs:	606					

In table 2.26 and 2.27, it was seen that there were 16 498 sequences in the original sample file but after QC with Mothur, there were 606 sequences left. Mothur created empty sequences (0 bp) with QC but these had to be removed when doing further analysis due to being shorter than 10 bp. The median length of sequences before QC was 522 bp and after QC was 193 bp. The longest sequences before QC were 785 bp long and after QC were 456 bp.

Table 2.28: Summary of the Rooibos commercial sample sequences. Before QC and primer trimming were done by Mothur so comparisons could be made between the different samples and pipelines.

MID10 before	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	33	33	0	3	1
2.5%-tile:	1	43	43	0	3	111
25%-tile:	1	406	406	0	5	1110
Median:	1	549	549	0	6	2219
75%-tile:	1	554	554	0	6	3328
97.5%-tile:	1	564	564	0	7	4327
Maximum:	1	758	758	3	8	4437
Mean:	1	459.287	459.287	0.0178048	5.43385	
# of Seqs:	4437					

Table 2.29: Summary of the Rooibos commercial sample sequences. After QC and primer trimming were done by Mothur so comparisons could be made between the different samples and pipelines.

MID10 after	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	-1	-1	0	0	1	2
25%-tile:	1	46	46	0	3	12
Median:	1	138	138	0	3	23
75%-tile:	1	218	218	0	4	34
97.5%-tile:	1	403	403	0	5	43
Maximum:	1	423	423	0	5	44
Mean:	0.863636	141.909	141.977	0	3.38636	
# of Seqs:	44					

In table 2.28 and 2.29, it was seen that there were 4 437 sequences in the original sample file but after QC with Mothur, there were 44 sequences left. Mothur created empty sequences (0 bp) with QC but these had to be removed when doing further analysis due to being shorter than 10 bp. The median length of sequences before QC was 549 bp and after QC was 138 bp. The longest sequences before QC were 758 bp long and after QC were 423 bp.

2.4.4.3) RDP

Table 2.30: Summary of the Honeybush natural sample sequences. After primer removal, before QC, with the RDP pipeline.

MID7	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	1	22	22	0	3	433
25%-tile:	1	112	112	0	3	4324
Median:	1	231	231	0	4	8648
75%-tile:	1	507	507	0	6	12972
97.5%-tile:	1	557	557	0	7	16863
Maximum:	1	606	606	6	11	17295
Mean:	0.993293	279.322	279.326	0.0141081	4.41208	
# of Seqs:	17295					

In table 2.30, it was seen that there were 17 295 sequences in the sample file. No files were able to be generated after QC with RDP because no 18S AM fungal gene was present in the

pipeline to choose as reference on the RDP website. The results shown here were the summaries of the sequences after primer trimming was done with Cutadapt. The median length of the sequences were 231 bp. The minimum length was 0 bp and the maximum length was 606 bp.

Table 2.31: Summary of the Honeybush commercial sample sequences. After primer trimming was done, before QC, with the RDP pipeline.

MID8	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	0	0	0	0	1	1
2.5%-tile:	1	37	37	0	3	255
25%-tile:	1	502	502	0	5	2544
Median:	1	531	531	0	6	5088
75%-tile:	1	549	549	0	6	7631
97.5%-tile:	1	564	564	0	6	9920
Maximum:	1	835	835	4	8	10174
Mean:	0.998231	461.037	461.038	0.0155298	5.33586	
# of Seqs:	10174					

In table 2.31, it was seen that there were 10 174 sequences in the sample file. The median length of the sequences were 531 bp. The minimum length of the sequences was 0 bp and the maximum length was 835 bp.

Table 2.32: Summary of the Rooibos natural sample sequences. After primer trimming was done, before QC, with the RDP pipeline.

MID9	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	7	7	0	2	1
2.5%-tile:	1	42	42	0	3	413
25%-tile:	1	225	225	0	4	4125
Median:	1	503	503	0	5	8250
75%-tile:	1	537	537	0	6	12374
97.5%-tile:	1	560	560	0	6	16086
Maximum:	1	778	778	11	9	16498
Mean:	1	390.589	390.589	0.0153352	4.99212	
# of Seqs:	16498					

In table 2.32, it was seen that there were 16 498 sequences in the original sample file. The median length of the sequences were 503 bp. The minimum length of the sequences was 7 bp and the maximum length was 778 bp.

Table 2.33: Summary of the Rooibos commercial sample sequences. After primer trimming was done, before QC, with the RDP pipeline.

MID10	Start	End	NBases	Ambigs	Polymer	NumSeqs
Minimum:	1	2	2	0	2	1
2.5%-tile:	1	42	42	0	3	111
25%-tile:	1	391	391	0	5	1110
Median:	1	529	529	0	6	2219
75%-tile:	1	548	548	0	6	3328
97.5%-tile:	1	563	563	0	7	4327
Maximum:	1	758	758	3	8	4437
Mean:	1	445.725	445.725	0.0178048	5.43047	
# of Seqs:	4437					

In table 2.33, it was seen that there were 4 437 sequences in the original sample file. The median length of the sequences were 529 bp. The minimum length was 2 bp and the maximum length was 758 bp.

From the QC results comparisons above, it can be seen that the QIIME and Mothur pipelines do give roughly similar results. The QC results of the RDP pipeline could not be generated seeing that there was no reference gene that could be used, but even if it could be generated, the first step of the pipeline was done against the fungal ITS gene, and this makes it unreasonable to use as a pipeline. It was thought best to rather look at the QIIME and Mothur pipelines in future until there is an adequate reference gene available on the RDP pipeline. The RDP pipeline was still used in further analysis (where possible) in chapter 3, but for future analysis it would be wise to not consider the RDP pipeline.

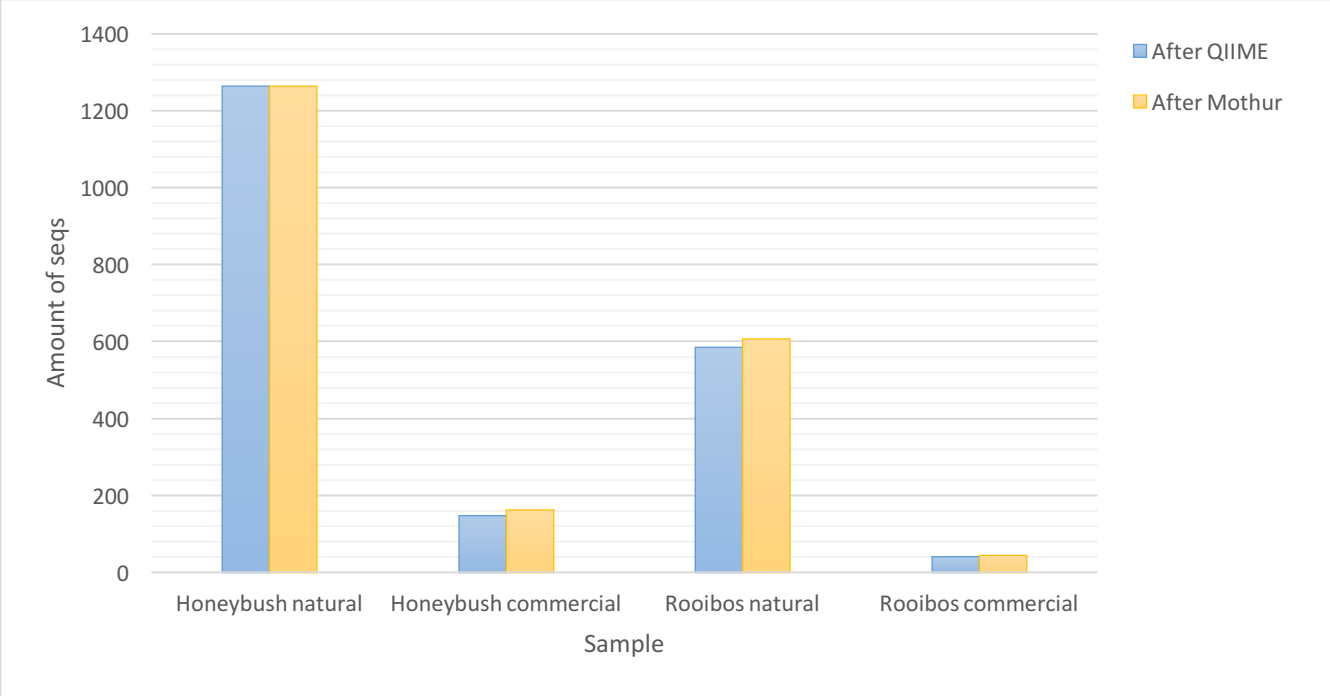


Figure 2.49: This figure shows the amount of sequences present in each sample after initial processing was done by QIIME and Mothur. The data of the RDP pipeline was not added to this figure, seeing that no feasible data was obtained after the initial processing step. In this figure it can be seen that the QIIME and Mothur initial processing steps had outputs that were likely to be the same.

In figure 2.49 only the amount of sequences were compared. It is possible to compare the other results in this section graphically, but it does result in relatively the same representation.

After the comparisons were done with the summary.seqs command, the Python script in Appendix V was used to determine the sequence identities, but the results were inconclusive (as was also noted in Appendix V).

2.4.5) NextGen Workbench

This method's results were discussed in Appendix III. This was not one of the standalone pipelines used in this study, but was rather another program that can be used to conduct initial processing on the sample reads in future studies.

Chapter 3: Classification and further processing of the sample reads

3.1) Overview

This chapter discusses the second stage of the pipeline analyses. The two major parts of this stage was the OTU picking, classification and phylogenetic tree generation, that is then

followed by measurement of the diversities within and between samples, and other statistical tests. In this chapter, it will be seen that there were many different obstacles. One major obstacle being that there is no database currently present that contains enough 18S AM fungal data that can be used in downstream analysis. Both the OTU picking and classification was thus a problem in this regard. Phylogenetic tree generation was able to be done – but not by using any of these pipelines.

3.2) Introduction

3.2.1) OTU picking, classification and phylogenetic tree generation

In the absence of a good reference database, or when the sequences are from unknown organisms, easy taxonomic assignment is not possible and the reads must be clustered into groups called operational taxonomic units (OTUs). MEGAN (Huson *et al*., 2007) and TANGO (Alonso-Alemany *et al*., 2011) databases cannot be used due to 18S AM fungal sequences being insufficient or not present. An OTU is an operational definition of a species or group of species often used when only DNA sequence data is available. During this phase, processing of the reads were tried in order to be able to draw comparisons between the samples. The first step was to cluster reads based on similarity into OTUs and to select a representative sequence for each OTU. Each OTU was then classified by comparison to a reference database and a phylogeny inference was made based on sequence alignment, but a phylogenetic tree was not able to be constructed due to insufficient taxonomic classifications made. EzTaxon-e (Chun *et al*., 2007) can facilitate the phylogenetic tree generation of the sequences which also occur in GenBank. Other databases that were looked at are mentioned in table 3.1. This table also contains the different chimera detection, aligner, tree building and alignment software used in correlation with these databases.

Table 3.1: The main reference resource databases as discussed and illustrated in Santamaria *et al*. (2012). The RDP and Silva 104 databases were looked at in this study.

Database	Chimera detection	Aligner	Tree builder	Alignment
RDP	Pintail	Infernal	RDP Classifier	Automatic

Greengenes	UCHIME ChimeraSlayer	Infernal	FastTree	Automatic
SILVA 104 and 108	Pintail	SINA	ARB Parsimony	Curated
EzTaxon-e	No	SINA	ARB Parsimony	Curated

3.2.1.1) Metagenomic databases for taxonomic assignment of fungi and AM fungi

The establishment of culture collections have caused breakthroughs in species identification. One such establishment is called INVAM (International Culture Collection of Vesicular Mycorrhizal Fungi) in North America. Various descriptions were found on the INVAM page that was provided by Prof Joe Morton and can be found at http://invam.caf.wvu.edu. A. Schüßler has made a list of all the know Glomeromycota and this list was found at http://www.lrz-muenchen.de/~schuessler/amphylo/amphylo_species.html (Rosendahl, 2008). The Silva 104 database containing sequences of various fungal 18S genes was used during QIIME analysis as well as the MaarjAM database which contained virtual taxa of AM fungi. Virtual taxa (VT) are designated as such at the time of the VT erection and also represents the VT. It should have a high quality and be as long as possible. If it is available, the VT sequences can be selected from sequences that were obtained from cultured or well identified spores (Opik *et al*., 2014).

3.2.2) Measure diversity and other statistical tests

OTU information (number of OTUs, abundance of OTUs) and the phylogenetic tree generated from phase 2 would have been utilized to estimate diversity within and between samples. Silva (Pruesse *et al.*, 2007) was used to generate phylogenetic classifications for the rRNA samples in the European Nucleotide Archive. Pintail was not used to screen anomalous sequences (Ashelford *et al*., 2005), because it was based on bacterial 16S data. Pintail is also used in the RDP pipeline. Additional statistical analysis to test the significance of the diversities can be done in future when this phase of the pipeline is successfull. In this study, alpha diversity (a measure of diversity within a sample) and beta diversity (a measure of diversity between samples) was the important factors that had to be determined in order to compare diversities. Chao1, Chao2, Jacknife1, Jacknife2 and Bootstrap richness estimators could also be calculated in future (Opik *et al*., 2009).

3.3) Methodology

3.3.1) QIIME

The QIME website had a tutorial that explained how to process 18S data (www.qiime.org/1.4.0/tutorials/processing_18S_data.html). This tutorial could also be used to analyze mixed 18S/16S data. Most of the steps described in this tutorial are identical to the standard 16S pipeline described on the QIIME website at http://qiime.org/tutorials/. Reference data might be required for OTU picking, taxonomic assignment and template-based alignment building. The QIIME code used, was explained in Appendix I.

The QIIME pipeline (Caporaso *et al*., 2010) worked with compatible versions of Silva 104 and 108 database releases. The Silva 104 database contained all the non-curated sequences, whereas the Silva 108 database contained curated sequences and had no duplicates. Both these databases contained data for all three domains of life and both were available at http://www.arb-silva.de/download/archive/qiime/. The Qiime_files_r104.zip was downloaded which contained the Silva 104 database sequences. This file contained several files that were required in the following steps to process the 18S AM fungal data.

The Silva 104 database was used as the reference dataset because it contained files that were specifically modified to help with the integration into the QIIME pipeline for marker gene analysis. UCLUST v1.2.22q was used for clustering the Silva 104 files. If necessary in future studies, Primer Prospector's (http://pprospector.sourceforge.net/) module *clean_fasta.py* can be used to remove spaces and convert "U" to "T" in the FASTA files. Core Silva aligned the sequences from the complete Silva 104 set and filtered to 80% identity with UCLUST (Edgar, 2010) which was followed by filtering out positions that were greater than 99% gaps. The version of QIIME at the time of this analysis, used FASTTREE 2.1.0, to construct phylogenetic trees.

The representative set used was generated by clustering the full Silva 104 release FASTA file at 97% identity. Taxonomy mapping files were generated using parsing strings from the Silva FASTA file. A RDP compatible file was generated with a custom parser in order to get 6 levels of taxonomy followed by hand curation in order to clean up the empty levels of the taxonomic definition. The representative sequences were filtered from the original Silva 104 alignment to remove positions that were > 90% gaps and entropy filtered to remove the 10% most entropic/variable positions. The resulting alignment was then planned to be the input into

FASTTREE in order to build a phylogenetic tree. The tree had to be manually rooted between the Archaeal and Eukaryotic clades. The reason for this is that there was a possibility that both Archaeal and Eukaryotic clades were present in the samples seeing that the samples were taken from soil, even though the primers were AM specific.

When, in future, wanting to use the Silva 108 release, it is important to take note of a few changes from the 104 release. In addition to the filtering steps taken for the 104 release, all the sequences that contained any degenerate characters have to be removed in the Silva 108 release. The RDP compatible mapping files for the family, genus and species levels have to be created for the full dataset and for eukaryotes alone. Also, a larger amount of memory is used for lower level taxonomic assignments.

In order to be able to run this pipeline with the RDP classifier, Java v1.6.0 or later, is needed as errors have been observed when using earlier versions.

There are four main steps in processing the 18S data with the QIIME pipeline:

- Picking Operational Taxonomic Units (OTUs)
- Assigning OTUs to taxonomic identity
- Separating OTU tables according to domain
- Generating and filtering an alignment to construct a phylogenetic tree

The information described in this section, is based on the information on the QIIME website (Caporaso *et al*., 2010), specifically that found on the QIIME tutorials page (http://qiime.org/tutorials/index.html).

3.3.1.1) Picking OTUs

The default settings of pick_otus.py were used and no different than that used with the 16S data. Sequences were clustered at 97% identity and new clusters were allowed. A reference based approach was used with the Silva dataset that discarded sequences that did not cluster with the reference set. The advantage of using this approach, as mentioned on the QIIME website, is that one can use the reference taxonomic assignments and tree rather than generate new ones but this may cause a large amount of sequences to be discarded.

By using the *pick_otus.py* script, OTUs were picked against the Silva 104 dataset and any clusters that did not match were discarded. This generated a text file as the OTU mapping file. The OTU table with taxonomy strings were built directly from the reference dataset (Caporaso *et al*., 2010).

3.3.1.2) Assigning OTUs to taxonomic identity

The script *pick_otus.py* was used to generate *de novo* OTUs which created a text file that could be used to make a representative sequence set for downstream analysis with the *pick_rep_set.py* script. Taxonomies were assigned to the *de novo* OTUs that were generated with the *assign_taxonomy.py* script. There was an initial problem with this step in that it required specific memory usage. When using *–rdp_max_memory* as an option with the RDP assignment method, specified with the *–m* option, then the memory usage was specified. A problem occurred when using the QIIME pipeline in the VM. The error on the VM was the following:

```
StdErr:
uclust v1.2.22q
(C) Copyright 2009-10 Robert C. Edgar
Licensed ONLY for use in PyNAST and QIIME
Killed 2.5Gb 60.1% Reading 607182 seeds
```

This error was an indication that there was not sufficient RAM available on the VM. What was also tried was to increase the allocated RAM to the VM, but this also failed and the error given was the same. The only feasible, and preferred, option by many QIIME experts on the QIIME forum (https://groups.google.com/forum/#!forum/qiime-forum) was to download the QIIME VM on a server and then run the command from there, which was done on the Rhodes University Bioinformatics Unit's (RUBi's) server.

If the RDP or UCLUST (Edgar, 2010) methods are not preferred by the user, then it will be wise to use the BLAST method as a method of assignment where the generic taxonomic mapping file can be used instead. An OTU table was built which included the taxonomic assignments (the UCLUST assignment was preferred with the 18S data).

3.3.1.3) Separating OTU tables according to domain

This step might be redundant because it is usually implemented when OTU tables from mixed 16S/18S samples needed to be separated. It was still wise to do this step in order to make sure that the OTUs continued with during downstream analysis, only contained the 18S data and no other data. The script used for this was *split_otu_table_by_taxonomy.py*. The output directory

contained the OTU table for eukaryotes which was used in further downstream analysis with QIIME.

3.3.1.4) Generating and filtering an alignment to construct a phylogenetic tree

This step was not able to be achieved seeing that the clusterings and taxonomic assignments made, contained very little information. This step was described in order to give information on how to go about this step in the pipeline when adequate results are calculated in previous steps. The module *align_seqs.py* could be used with the Silva 104 reference set to create an alignment where the core Silva aligned set will be used as the template. The representative set FASTA file created with the OTU picking step above, can be used. The alignment can then be filtered. When analyzing 16S datasets, a Lanemask (Caporaso *et al*., 2010) is usually applied in order to remove high entropy positions, but QIIME has incorporated a dynamic entropy and gap calculation with the *filter_alignment.py* module that removes the need for a Lanemask when working with 18S data. The default is to use the parameters *–e 0.10* and *–g 0.80* which specifies that the 10% most variable positions and positions greater than 80% gaps will be removed. After this alignment, a tree can be built using *make_phylogeny.py*. The trees the OTU table created can then be used in downstream QIIME analysis, as can be seen in the tutorial on www.qiime.org/1.4.0/tutorials/tutorials.html#view-statistics-of-the-otu-table .

3.3.2) RDP pipeline

3.3.2.1) Classifier

The classifier was limited to 50 000 sequences per sample file (Wang *et al*., 2007). The files as output from the initial processing of the reads were used as input for this module. The genes that were able to be selected to do classification against were not 18S genes. The genes used to do classification against were the “Warcup fungal ITS trainset 1” and “UNITE fungal ITS trainset 07-04-2014”. The confidence cutoff was set to be 80.

3.3.2.2) Aligner

The aligner module was limited to 1 000 000 sequences per sample file. This tool aligns processed and trimmed sequences using the fast, secondary structure aware, Infernal aligner (Nawrocki *et al*., 2009). The bacterial and archaeal models have been recently updated to use larger training sets with broader phylogenetic coverage. The Infernal aligner had been updated to Infernal version 1.1rc4 that, compared to the previous version used in RDP 10, allowed better optimization of the Infernal parameters to provide improved handling for partial sequences. It is also reported to be 7.5 x faster.

It does have several different limitations to it. The sequences must only contain IUPAC nucleotide codons, gap characters will be removed, and sequences with other characters will be ignored.

The alignment could not be done correctly seeing that there was no 18S AM fungal gene to do alignments against.

3.3.2.3) Complete linkage clustering

The clustering tool was limited to 150 000 sequences per sample file. Clustering of sequences had to be done by the complete-linkage clustering method. When the clustering job was complete, an email was sent to the address that was specified. This email contained a link to download the clustering results. The maximum distance was set to 15% and the step size was set as 1.5. The step size referred to the increment between the clustering distances.

Due to the alignment step that could not be done, no clustering was able to be completed.

3.3.2.4) Chimera check

Chimera check is a very important step of an analysis pipeline. Checking an amplicon sequence file for chimeras using USEARCH 6.0 was done. This tool was limited to 60 000 sequences per sample file through the website http://fungene.cme.msu.edu/FunGenePipeline/chimera_check/form.spr.

USEARCH was run in the UCHIME *de novo* mode which meant that this tool would only work with experimental data that contained both chimeric and non-chimeric sequences which were in higher abundance (Edgar *et al*., 2011).

3.3.3) Mothur

The Mothur tutorial, that can be found at http://www.bio.utk.edu/fesin/FESIN2010/Workshop2010/Amend/Mothur_tutorial.pdf, was used to analyse the 18S AM fungal data. The Mothur code used was explained in Appendix II.

Mothur (Schloss *et al*. 2009) was specifically designed to work with environmental sample reads that can be reasonably aligned and which are also useful for distinguishing species. The 16S in prokaryotic systems seemed to fit both of these criteria, but fungi did not. The fungal

ITS region can be used to distinguish between species, but it was still difficult to get a meaningful alignment across the kingdom Fungi. The other loci like the 18S or 28S, could yield reasonable multiple sequence alignments but they were of no use when determining OTUs depending on the groups of fungi being studied. The parts of Mothur that can deal with clustering of OTUs based on the multiple sequence alignments would probably not work no matter which locus is being used. There are a number of programs which can use this pairwise alignment, for example Blastclust, CD-HIT, and UCLUST. These types of clustering algorithms were integrated into this tutorial to enable the user to work with programs like Mothur. The tutorial was divided into two sections called "Phylogenetic Analysis" and "OTU Analysis". In the phylogenetic part, Mothur was used on the raw 454 data of the 18S samples (as explained in Chapter 2). The OTU analysis part uses CD-HIT-EST to calculate OTUs based on 97% ITS sequence identity, which needed to be changed to look at 97% 18S identity.

3.3.3.1) Unique sequences

When working on a system that possibly has a low relative complexity, the samples may be dominated by a few species. In order to save hard drive space and computational effort, Mothur collapsed all the identical sequences in a .names file which generated a list of unique sequences. The **unique.seqs** command was used, which worked on identical sequences. This meant that redundant sequences must have exact nucleotide matches and sequence lengths.

3.3.3.2) Align sequences

Mothur has a built-in aligner. It had both advantages and disadvantages. One of the advatages was that it could process large amounts of data relatively rapidly and produce alignments that were easy to use in downstream Mothur analyses. A disadvantage was that Mothur needed a template alignment to work against. Because the 18S sequences were used, an alignment needed to be created. This was done by using the MaarjAM (Opik *et al*., 2010) database and MAFFT (Katoh *et al*., 2002; Katoh and Standley, 2013) as multiple sequence alignment (MSA) algorithm. The command used here was **align.seqs** where the candidate was specified as the FASTA file that was the output from the unique.seqs command, and the template was specified as the newly aligned MaarjAM database. "flip=T" was added to the command in order to take care of any reverse complimentary sequences that were found in the alignments.

3.3.3.3) Chimera check

The chimera check step in the Mothur pipeline worked by comparing the closest match of the first half of sample sequences to the template alignment, and the closest match of the other half

of the sample sequences to the template alignment. If there were to be two different matches, then the sequence will be classified as chimeric. There was also a chance of some false positives being generated. If the dataset is small enough, in future it would be wise to check these by hand. The default settings for parameters were used during this step.

Chimeric sequences were removed from the samples by using the **remove.seqs** command. These sequences had to be removed from the groups, name and alignment files in order for everything to match up.

The command used to do chimera checking was **chimera.slayer**. Both the FASTA file produced as output in the previous step and the template, was used as parameters. In order to remove the chimeric sequences from the files, the remove.seqs function was used. The parameters for this function was the name, groups and alignment files. The **summary.seqs** command was used again in order to determine how many sequences were chimeric. Because the alignment contained a lot of gaps, the **filter.seqs** command was used again in order to get rid of all the alignment positions that only contained gap characters.

3.3.3.4) Produce a phylogenetic tree

The programs Jalview (Waterhouse *et al*., 2009) or MEGAN (Huson *et al*., 2007) can be used in order to generate a phylogenetic tree from alignment files, but in this case only Jalview can be used since MEGAN does not have any 18S AM fungal sequences present in its database.

When working with large datasets, the program ClearCut (Sheneman *et al*., 2006) can be used. For Clearcut there is a wrapper included in Mothur. FASTTREE (Price *et al*., 2010) can also be used, but it was important to note that the trees generated by this program are not easily compatible with Mothur.

3.3.3.5) Compare community phylogenetic structure

Mothur offers three types of analysis to be done after the phylogenetic trees were constructed. Parsimony (Goloboff, 2003) , Unifrac (Lozupone and Knight, 2005) and Libshuff (Schloss, 2008) have been integrated into Mothur. Each of these analyses offer variations on how to test whether the phylogenetic community structure in one sample differs from the other samples.

Parsimony test generates random trees from a pool of species and then compares the number of changes in the tree topology accounting for the differences among communities against the randomized null. The tree is read into memory along with the group and name files. It can then be tested whether any of the samples differ from random structure. The number of randomizations was reduced from 1 000, which is the default, to 100 in order to reduce the running time.

The functions used here are **parsimony**, which determines the number of randomizations; and **read.tree** to read the tree, groups and name files into memory. This will produce a significance score which can be used to tell whether the community structure of at least one sample differs from the random. Pairwise comparisons of these samples can also be looked at. This step can be computationally intensive, because it increases exponentially with the number of samples. The parsimony command will give an output summary file in which all the pairwise comparisons can be seen. The statistical aspect of this step is to measure the significance and should not be interpreted as the degree of dissimilarity.

UniFrac is another metric of community phylogenetic dissimilarity. There are two versions of this program. One being the weighted version and the other being the unweighted version. With the weighted version, the abundance of sequences is taken into account along with the phylogenetic similarity, or the shared branch length, in order to calculate the similarity between communities. The unweighted version does not take the sequence abundance into account (Lozupone and Knight, 2005).

Either the function of **unifrac.weighted** or **unifrac.unweighted** can be used. This will result in a p-value that indicates how many of the samples differ from the randomly generated tree. The groups can then be compared to each other and a distance matrix can also be generated. The output will be the pairwise differences between the communities and the ".dist" file will give information in a distance matrix which can be used for ordinations or multivariate statistics.

There is also a built-in function which can be used to generate principal component (PCoA) ordinations using the distance matrices and the **pcoa** function. This will lead to a scatter plot (by only using the first two of the three axes) being produced and also a "loadings" file that can be used to determine how much variance was described by each of the axes.

3.3.4) Generating phylogenetic trees with Jalview and MEGA7

After failure of most of the pipelines during the second stage of the sample read analyses, it was thought appropriate to generate phylogentic trees separately to be able to get an idea of the diversity within the different samples.

The sample files used in this case was that of the samples where the primers were taken out separately from a pipeline by using Cutadapt, and after QC analysis was done with Mothur. Mothur results was chosen as the QC results of choice, in this case, seeing that the results after QC for both the QIIME and Mothur pipelines were found to be relatively similar (section 2.4.4).

3.3.4.1) Generating phylogenetic trees with Jalview

The analyses were done in Jalview (Waterhouse *et al*., 2009) 1.0 for Mac OS. After QC was done with Mothur, and the resulting files were all FASTA files, these files were used. Firstly, the sequences that were shorter than 10 bp were deleted from the files. But when aligning and doing phylogenetic tree generation with the sequences that were from 11 bp onwards in length, the alignments were done very poorly and it was not possible to generate phylogenetic trees. Numerous different alignment parameters were tried in order to achieve better alignments and also to generate the phylogenetic trees for each sample file. As a drastic measure, all sequences shorter than 100 bp were deleted from all the FASTA files. The multiple sequence alignments were done with MAFFT. Even before MAFFT alignments were done and the sequences were sorted by pairwise identity, it was clear that some of the sequences were very close to being identical by using the nucleotide colour scheme in Jalview. The MAFFT alignments were done by using the default parameters in Jalview and the phylogenetic trees were generated by using neighbor joining using DNA.

3.3.4.2) Generating phylogenetic trees with MEGA7

Phylogenetic trees were also created with MEGA7 on a Mac OS. The output files of the Mothur QC steps were again used. The files were first individually viewed in Jalview where the nucleotide colour scheme was again used. Then, the sequences were sorted according to pairwise identity and all the sequences that were shorter than 100 bp were taken out. Sequences that were clearly not close to having identical bases, were also taken out. This lead to, at most, 100 sequences being left in each of the sample files. The amount of sequences were decreased

to roughly 100 sequences per file in order to decrease the run time of the phylogenetic tree analysis.

After the sequences were prepared, multiple sequence alignments were run by using MAFFT (with default settings) in Jalview. The alignment files were saved as FASTA files and converted to .meg files in MEGA7. The phylogenetic trees were constructed by using maximum likelihood calculations. The maximum likelihood trees were calculated by using the bootstrap method as the test of phylogeny with 500 bootstrap replications, the substitution type was set to be nucleotides with the model being the Kimura 2-parameter model, the rates among sites were set to be gamma distributed (G) with the number of discrete gamma categories set as 5, all the sites were to be used with the maximum likelihood tree inference model set to use nearest-neighbor-interchange (NNI) with the initial tree being a neighbor joining tree, and lastly, the number of threads were set to be 1.

The reason for using maximum likelihood to construct the phylogentic trees were because it is a statistical method of estimating unknown parameters of a probability model. A parameter, in this case, is seen as a descriptor of the model. In phylogenetics there are many different parameters including rates, differential transformation costs, the tree itself. The parameter in this case is a quantity proportional to the probability of observing the data given the model of P(D|M). The maximum likelihood tree examines the likelihood function to determine where it is greatest, as well as the value of the parameter interests (the branches and branch lengths) at a specific point (adapted from “Principles of Phylogenetics”, University of California).

3.3.5) Optional methods

There are various different methods that were tested during this study. Some of the methods were studied in order to achieve results for picking OTUs and classifying the sequences after multiple sequence alignments. If adequate results are achieved with these methods, diversity statistics can be done with, for example, R and RStudio. The appendices that contain information regarding the other methods are appendices III, IV, V and VII.

3.4) Results and Discussion

3.4.1) QIIME

3.4.1.1) Picking OTUs

When picking OTUs with the *pick_otus.py* module, sequences were clustered at 97% identity and new clusters were allowed to form. A reference based approach was used with the Silva

104 dataset. This approach caused the discardment of sequences that did not cluster with the reference set and lead to only two sequences being clustered. The advantage, as previously mentioned of this approach, is that one can use the reference taxonomic assignments and tree rather than generate new ones, but this may cause a large amount of sequences to be discarded. This was not seen as an advantage in this case, but rather a disadvantage because of the high sequence losses.

The *pick-otus.py* script was used to pick the OTUs against the Silva 104 dataset and the clusters that did not match, were discarded. A text file was generated that contained the OTU information, and the OTU table with taxonomy strings were built from the reference dataset. The text file that was created for the picked OTUs did contain multiple picked OTUs from each of the sample files, but the problem came in when assigning taxonomies to the various picked OTUs. The text file had the following format:

```
denovo0 SeqID
denovo1 SeqID SeqID SeqID
denovo2 SeqID
…
```

The ‘de novo’ is a mention that *de novo* OTU picking was used because, when using the reference dataset, no/very little results were obtained. The *de novo* OTUs are numbered from zero onwards, and this is followed by the sequence IDs of the sequences that were clustered into that specific OTU. Many different sequences can be clustered into a specific OTU when the sequences might have the same evolutionary ancestor.

3.4.1.2) Assigning OTUs to taxonomic identity

The module *pick_otus.py* was also used to generate the *de novo* OTUs which created a text file that was used to make a representative sequence set for downstream analysis with the *pick_rep_set.py* module. To assign the taxonomies to the *de novo* OTUs that were generated, the *assign_taxonomy.py* module was used. There was a problem with this step in that it required specific memory usage. The option *–rdp_max_memory* was used with the *–m* parameter in order to be able to specify the memory usage, but then even when the memory usage was decreased to 3000 MB, the same error came about. Fewer memory usage than this, completed this step, but it then took quick some time longer. A problem occurred when using the QIIME pipeline in the VM and so QIIME was also installed on a Mac OS X 10.10.2 and the RUBi server.

The results for the taxonomic assignments were also in a text file and had the same format as that of the OTU picking output text file, but instead of the sequence IDs, the sequences were classified taxonomically. The problem here was that all the sequences in all the samples were unable to be assigned to any of the AM fungal species, and was mentioned to be 'unassigned'. The only assignments that were able to be made was that of a few bacterial species, for example:

```
denovo0 Unassigned
denovo1 k__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__; f__; g__; s__
denovo2 Unassigned
…
```

The 'k', 'p', 'c', 'o', 'f', 'g', 's' are the kingdom, phylum, class, order, family, genus and species respectively.

If the RDP or UCLUST methods are not preferred in future analysis, then it will be possible to use the BLAST method as a method of assignment where the generic taxonomic mapping file can be used instead. An OTU table can be built which will include the taxonomic assignments, but the UCLUST assignment is preferred with the 18S data.

3.4.1.3) Separating OTU tables according to domain

This step might not be necessary when working on these four samples, but this is usually implemented when the OTU tables from mixed 16S/18S samples need to be separated. In this study, it was thought to be appropriate to do this step to make sure that the OTU tables continued with during the study, was only that of 18S data. The module used for this step was the *split_otu_table_by_taxonomy.py*. The output directory then contained the OTU table for the fungi with which downstream analysis could be done in the QIIME pipeline. But in this output OTU table file, there were only two sequences' classification present. Both the OTUs were classified as being from AM fungi This indicated that the Silva 104 dataset used as the reference set did not contain enough AM fungal sequences.

This step was also tried against the MaarjAM database, but errors kept occurring seeing that the code from the QIIME pipeline specifically identified the sequence IDs in the Silva dataset and the identifiers in the MaarjAM database was unique to that dataset and thus did not agree with any other databases such as GenBank. The reason for the highly unique sequence

identifiers in the MaarjAM database is that these sequences are from virtual taxa. A way around this problem will need to be found during future studies.

3.4.1.4) Generating and filtering alignments to construct phylogenetic trees

The module align_seqs.py was used with the Silva 104 dataset to create the alignments where the core Silva aligned set was used as the template. The representative set FASTA file created previously with the OTU picking step above, was used. The alignments were then filtered. The QIIME pipeline has incorporated a dynamic entropy and gap calculation with the filter_alignment.py module that removed the need for a Lanemask as that being used when working with 16S data. The default used the parameters *–e 0.10* and *–g 0.80* that specified that 10% of the most variable positions and positions greater than 80% gaps, were removed. After this alignments, a phylogenetic tree can be built for every sample using the module make_phylogeny.py. This was not able to be completed because after the OTU picking step was completed, there were not enough sequences present (only two sequences left) in the FASTA file to use to do the alignment with and create the tree for each of the samples. When, in future, this step can also be completed, the trees can be used in further downstream QIIME analysis as can be seen in the QIIME website at www.qiime.org/1.4.0/tutorials/tutorials.html#view-statistics-of-the-otu-table .

QIIME did not work as well as expected because in the alignment step against Silva 104, very little taxonomies were identified. With the .tree file created, only two branches were able to be recognized and this cannot be used in further downstream analysis in the QIIME pipeline to do diversity studies.

3.4.2) RDP pipeline

3.4.2.1) Classifier results

This classifier in the RDP pipeline used bootstrapping as a statistical analysis method for analyzing the classification results. The classifier module used in the RDP pipeline is limited to 500 000 sequences. The UNITE fungal ITS trainset was used as the reference dataset seeing that no AM fungal dataset was present. It was interesting to see that the sequences had hits against the UNITE trainset seeing that the reads in the samples were from the 18S gene and that from the trainset were the ITS genes.

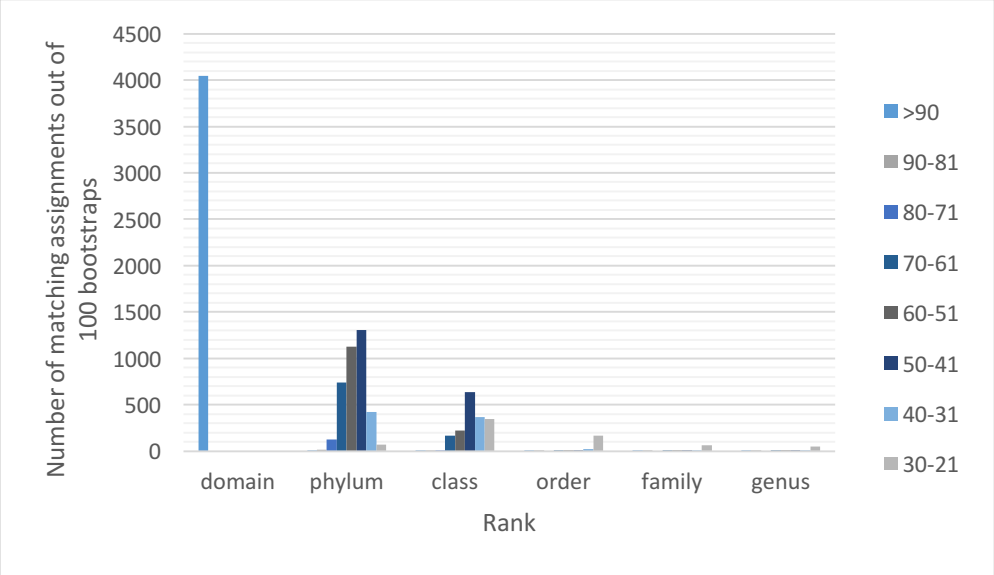


Figure 3.1: The bootstrap results for classification of the Honeybush natural sample reads. In this figure it was seen that the classification of the AM fungi became less as the classification went more in depth, and with a lower number of matching assignments out of the 100 bootstraps. The classification was done correctly to classify the sample as fungi, but the classification became less as the level decreased.

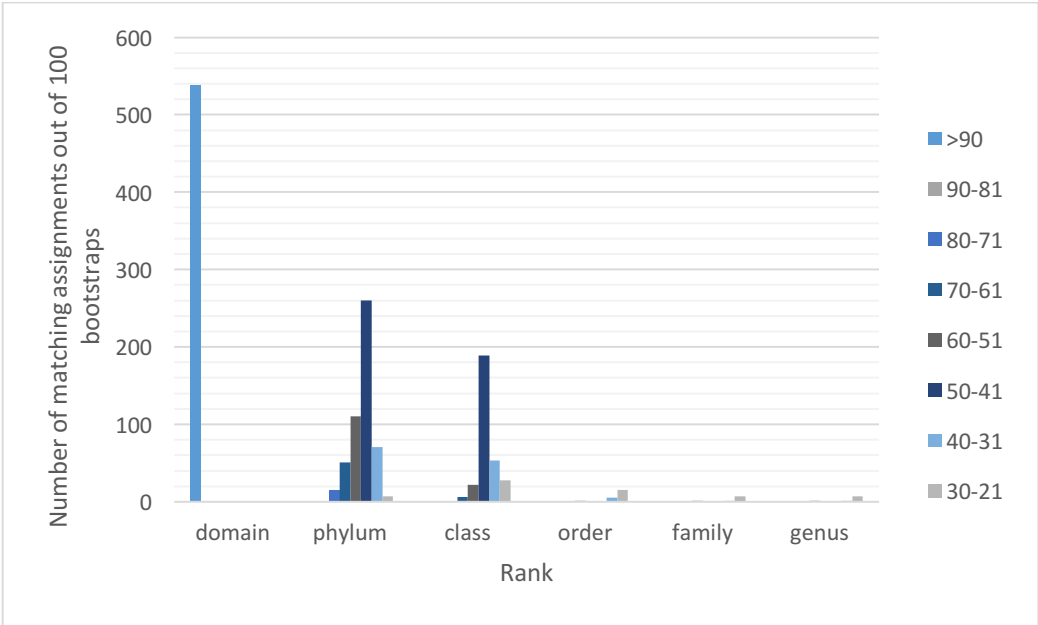


Figure 3.2: The bootstrap analysis results with classification of the Honeybush commercial sample reads. The classification became less as the classification level decreased out of the number of matching assignments of 100 bootstraps. The domain Fungi was correctly classified, but from phylum onwards, the classification was done for less and less sequences.

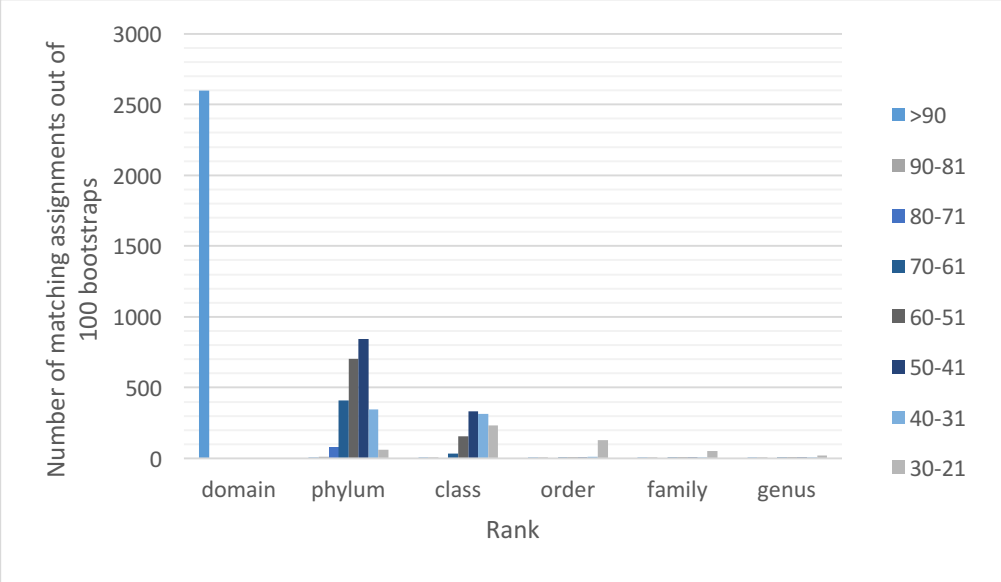


Figure 3.3: Bootstrap analysis results with classification of the Rooibos natural sample reads. The sample reads were less classified as the rank of classification decreased from phylum onwards. The domain Fungi was correctly classified.

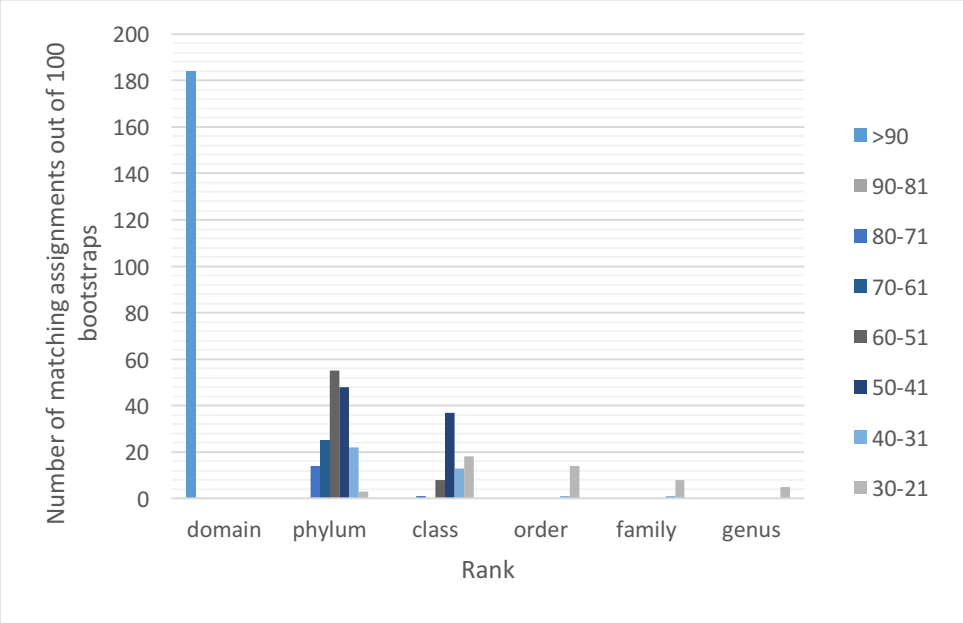


Figure 3.4: The bootstrap analysis results with classification of the Rooibos commercial sample reads. The classification decreased for each rank vs the number of matching assignments out of 100 bootstraps as the classification went from domain onwards. All the sequences were recognized to be in the domain Fungi.

From figure 3.1 to 3.4, it is clear that a proper reference dataset is needed in order to do proper taxonomic assignments against. The 18S reference dataset will only come with time as more research on AM fungi is done and more sequences are uploaded into AM fungal databases.

3.4.2.2) Aligner results

The aligner module was limited to 1 000 000 sequences per sample file. This tool aligns processed and trimmed sequences using the fast, secondary structure aware, Infernal aligner (Nawrocki *et al*., 2009). The bacterial and archaeal models have been recently updated to use larger training sets with broader phylogenetic coverage. It does have several different limitations to it. The sequences must only contain IUPAC nucleotide codons, gap characters will be removed, and sequences with other characters will be ignored.

The alignment could be done seeing that there was no 18S reference gene to align against. In order to overcome this problem, the multiple sequence alignments were done against the MaarjAM database using Mothur (Schloss *et al*., 2009) and the **align.seqs** command. In order to be able to feed this aligned sequence files into the RDP pipeline, the files had to be edited to contain the tag ">#=GC_RF
XXX...XXX.XX.....XXXX...."

Where "X" is the alignable position and "." is the non-alignable position. But after speaking to Benli Chai, which is part of the RDP staff team (rdpstaff@msu.edu), he also confirmed that the RDP pipeline does not have any 18S capabilities as of yet and research is currently going into developing 18S options which can also be used. But they are not putting in much effort as to the addition of AM fungal 18S databases seeing as too little sequencing information is currently available.

3.4.2.3) Complete linkage clustering results

The clustering tool was limited to 150 000 sequences per sample file. Clustering of sequences had to be done by the complete-linkage clustering method. The maximum distance was set to 15% and the step size was set as 1.5. The step size referred to the increment between the clustering distances. Clustering was not able to be done as the RDP pipeline did not recognize the identifiers used in the MaarjAM database, even with the tag added in the alignment step.

3.4.2.4) Chimera check results

Chimera check is a very important step of this pipeline. Checking an amplicon sequence file for chimeras using USEARCH 6.0 was done. USEARCH was run in the UCHIME *de novo* mode which meant that this tool would only work with experimental data that contained both chimeric and non-chimeric sequences which were in higher abundance (Edgar *et al*., 2011). This step was able to be completed and the non-chimeric sequences were saved in a separate FASTA file. This file can be used in further analysis with other analysis tools (as mentioned in

the appendices).

3.4.3) Mothur

3.4.3.1) Unique sample sequences

When working in a system with low complexity, the samples may be dominated by a few species. This found to be the case in the four different samples. This was found to be true especially for the sequences that had a longer length and both primers attached at the start of the analysis. The only problem with this was that the sequences could not be mapped to a certain type of AM fungi because the alignments failed in the following step.

In order to save hard drive space and computational effort, Mothur collapsed all the identical sequences in a '.names' file and generated a list of unique sequences. This command also only worked on identical sequences. Redundant sequences had to have exact nucleotide matches and also sequence length. This did not help reduce the file size, but the '.names' file would be useful in future analysis.

3.4.3.2) Align sample sequences

Mothur has a built-in aligner that has both advantages and disadvantages (as mentioned in section 3.3.3.2). Mothur also produces alignments that are easy to use in further Mothur analysis, but it requires a template to align against. Because there were no specific 18S AM fungal database available on the Mothur website to work against, the MaarjAM virtual taxa database for AM fungi was used and aligned using MAFFT (default settings). This MaarjAM alignment was used as the template alignment. The empty column or the positions downsteam of the target locus was removed from this template file. The flag 'flip=T' was used to include the reverse compliment in the alignments.

3.4.3.3) Check for Chimeras

Chimera checking was done by comparing the closest match of the first half of the sample sequeses to the template alignment, and the closest match of the other half of the sample sequences to the template alignment. If there were two different matches, the sequences were probably chimeric. The default settings were used to do chimera checking. A summary of the sequences were again compiled after possible chimeras were removed in order to determine how many of the samples' sequences were chimeric. On average four sequences per sample were found to be chimeric. These sequences were removed from the samples and the non-chimeric sequences were used in further analysis.

If the alignments did not start at position zero of the sequences, the gaps can be trimmed by using the filter.seqs command to get rid of all the alignment positions that contain gaps. Because the alignments were not good in this analysis, when removing the gaps, most of the sequences were removed from the files and very short sequences were left to do analysis against. In the tutorial previously mentioned, it stated that about 70% of the alignment was reduced, but in this case, between 80%-90% of the alignments were reduced.

3.4.4) Generating phylogenetic trees

After failure of most of the pipelines during the second stage of the reads analyses, it was thought wise to generate phylogentic trees separately from all the pipelines. The sample files used in this case were that of the samples where the primers were taken out separately from a pipeline and after QC analysis was done with Mothur. Mothur was chosen as the QC program of choice seeing that the results after QC for both the QIIME and Mothur pipelines were found to be similar (section 2.4.4).

3.4.4.1) Generating phylogenetic trees with Jalview

Phylogenetic trees were first constructed with Jalview. This was seen as an easy 'go-to' method, because results can be obtained quickly and easily. Table 3.2 contains a short summary of the samples' sequences used to construct the phylogenetic trees with neighbor joining. The table is followed by the four samples' neighbor joining trees.

No MOTU's were used in this case, but I do agree that the trees can still be useful for relatedness studies as well as determining the degree of diversities of the four different samples.

Table 3.2: Summary of the sequences before alignment with MAFFT with the default parameters in Jalview. These sequences were used to generate the phylogenetic trees for each sample.

Sample	Number of sequences	Minimum sequence length (bp)	Maximum sequence length (bp)	Average length (bp)
Honeybush natural	820	101	451	215
Honeybush commercial	77	107	516	222
Rooibos natural	420	100	456	244
Rooibos commercial	25	108	423	220

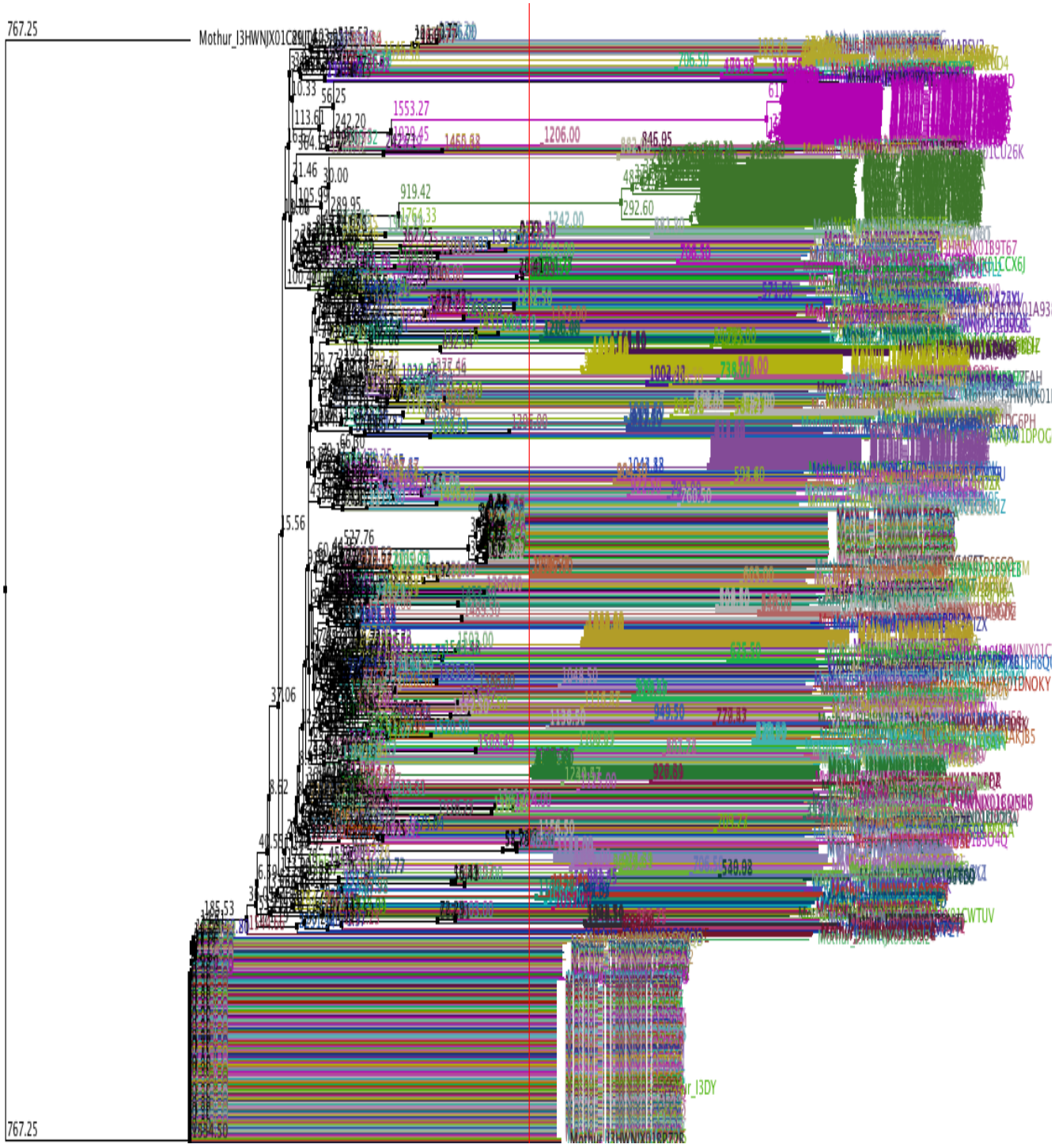


Figure 3.5: Phylogenetic tree generated in Jalview for the Honeybush natural sample. Neighbor joining was used to construct this phylogenetic tree after multiple sequence alignments were done using MAFFT in Jalview.

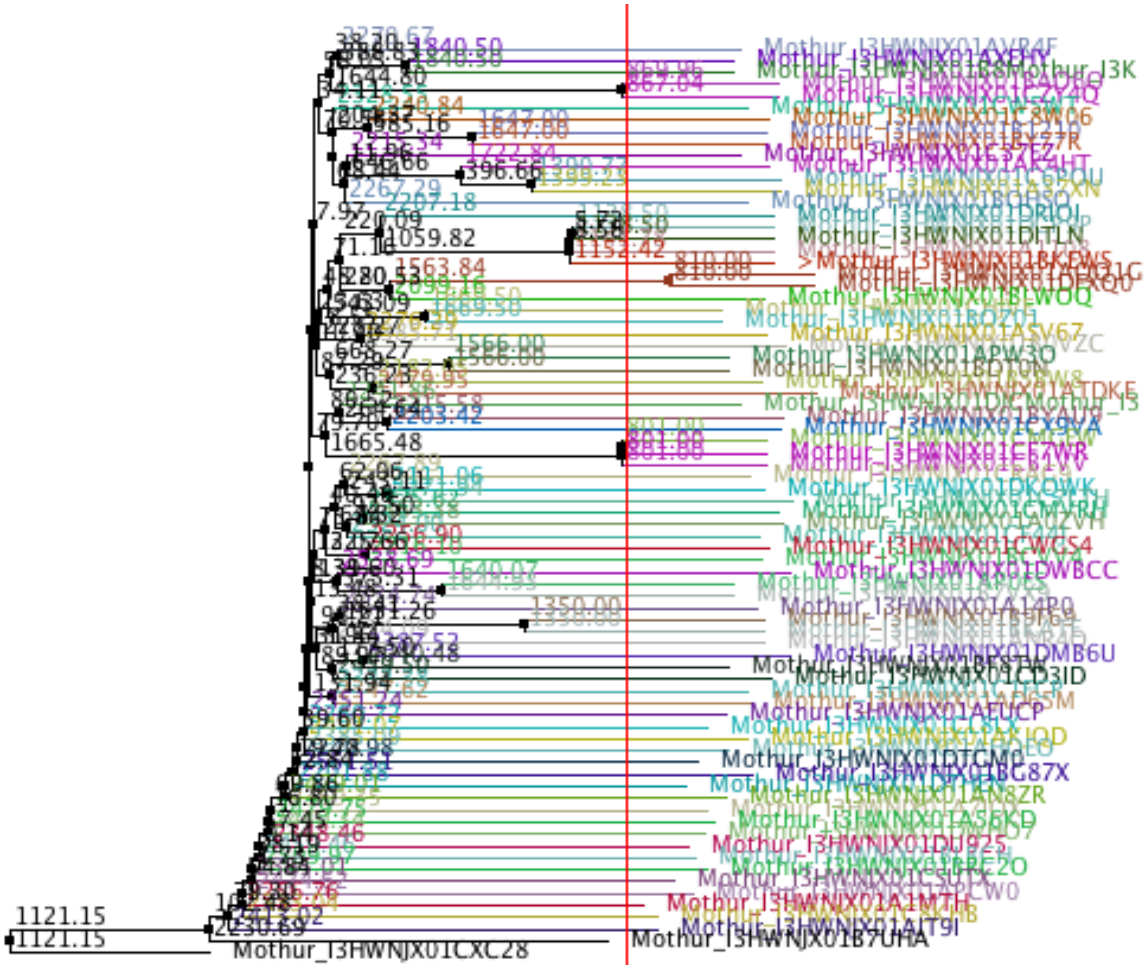


Figure 3.6: Phylogenetic tree generated for the Honeybush commercial sample. Neighbor joining was used to generate this phylogenetic tree after multiple sequence alignments were done with MAFFT in Jalview.

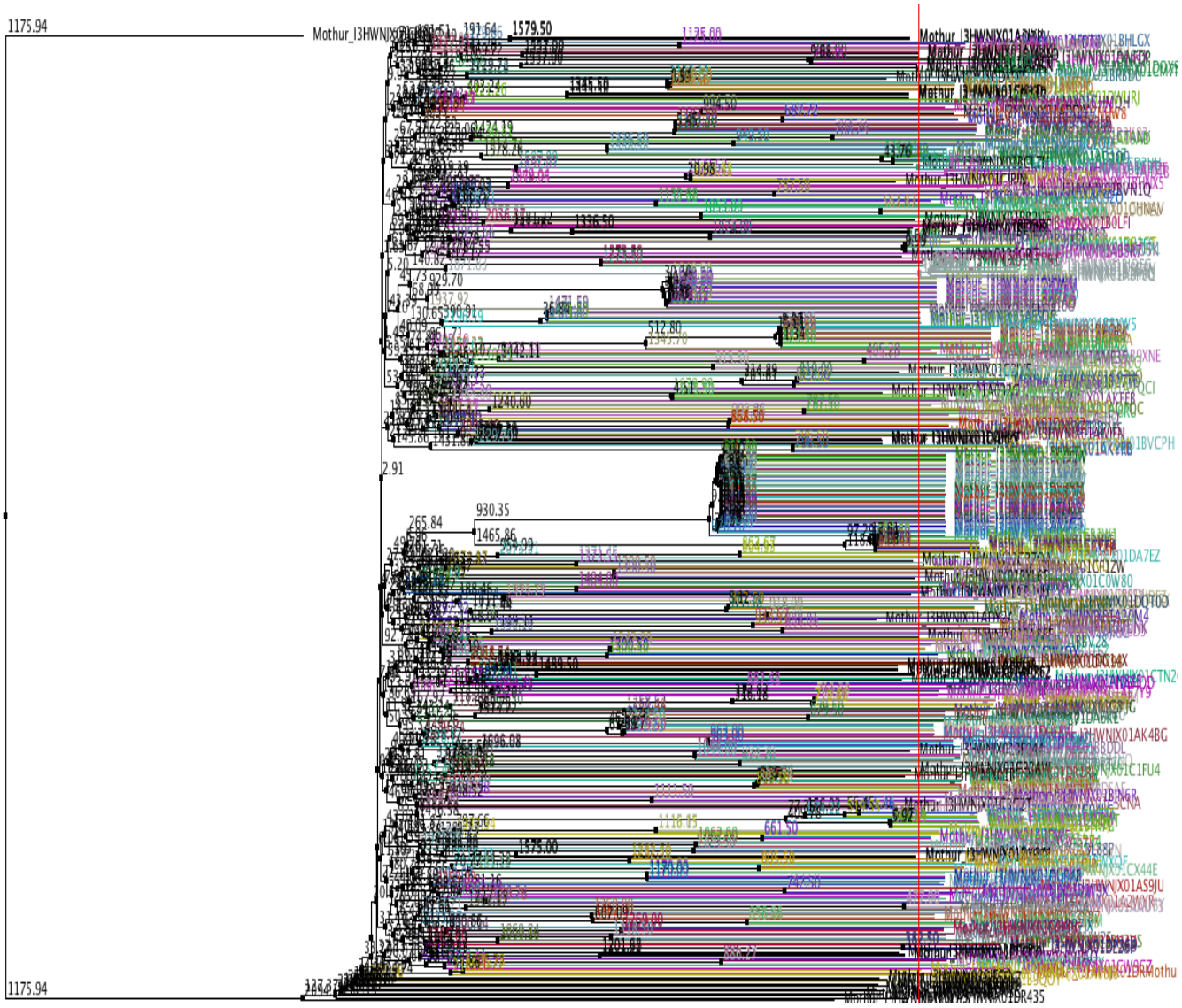


Figure 3.7: Phylogenetic tree generated for Rooibos natural sample. This tree was constructed using neighbor joining after alignments were done with MAFFT in Jalview.

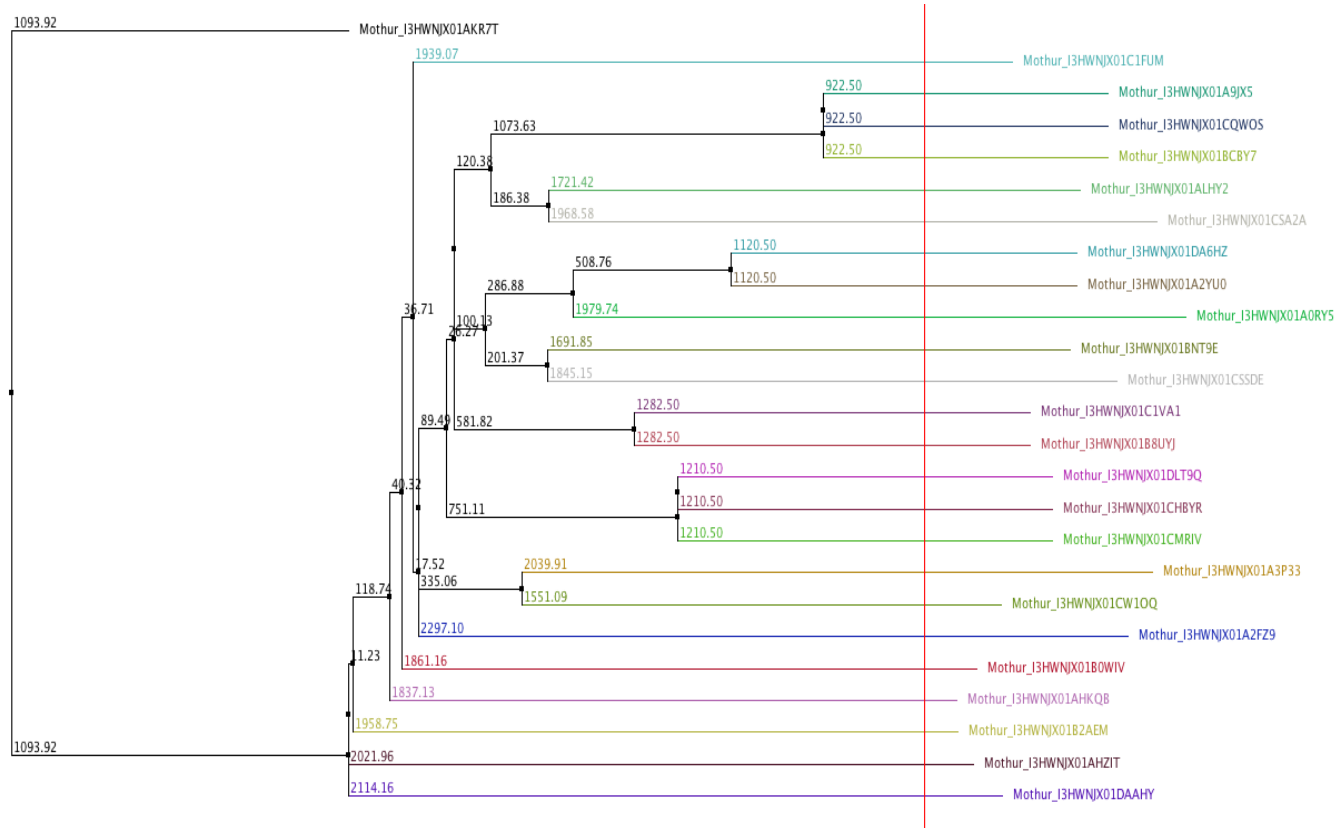


Figure 3.8: Phylogenetic tree generated for the Rooibos commercial sample. It was constructed after the multiple sequence alignment was done using MAFFT and using neighbor joining for tree generation.

The multiple sequence alignments were done with MAFFT in Jalview with the default parameters for MAFFT. In figure 3.5 to figure 3.8 it is clear that all the samples, both natural and commercial, had two major branches. The distances between the branches did differ from sample to sample, but figure 3.6 of the Honeybush commercial sample showed that the distances between its two major branches were very little. The amount of branches in the natural samples were a lot more than that in the commercial samples. This was due to the natural samples starting off with a lot more reads directly after pyrosequencing was done. This just shows that there is a higher diversity in the natural samples than in the commercial samples. This did agree with the hypothesis and literature (van der Heijden *et al*., 1998) where it was stated that the natural samples were expected to have a higher diversity of AM fungi within its samples.

3.4.4.2) Generating phylogenetic trees with MEGA7

This lead to the construction of phylogenetic trees with MEGA7 in order to have a clear comparison between the different samples. The reads in the four different samples were decreased to an amount that MEGA7 would be able to analyse in a short amount of time. Less

than 100 sequences were left in each file, except for the Rooibos natural sample that had 204 sequences because of high identities between many sequences in Jalview. It was decided that it would be interesting to determine the evolutionary history of the samples in order to get a clearer understanding of the diversity within these different samples, and so the phylogenetic trees in this section, were calculated by using maximum likelihood.

When using maximum likelihood, standard statistical techniques are used to infer probability distributions to assign probabilities to particular possible phylogenetic trees. It requires a substitution model for the assessment of probabilities of particular mutations. For example, when a tree has lower probabilities, it can be said that the tree requires more mutations at interior nodes in order to explain the observed phylogeny. Maximum likelihood is almost similar to maximum-parsimony, but it allows additional statistical flexibility. This flexibility then allows differing rates of evolution across the lineage and sites. Maximum likelihood requires that the evolution at the different sites and lineages has to be statistically independent. This is why maximum likelihood was chosen for analysis in this study – because it is suited to analyse distantly related sequences and it searches all possible combinations of tree topology and branch length. The disadvantage of this method is that it is computationally expensive to perform.

Phylogenetic methods rely on a substitution model that deals with the relative rates of mutation at various sites along the gene or amino acid sequences being studies. These models correct the differences in the rates of transitions and transversions in nucleotide sequences. The use of these models is necessitated by the genetic distances between the two sequences that increase linearly only for a short time after the sequences diverge from each other. The longer the time is after the divergence of the sequences, the more it is likely that the mutations occur at the same nucleotide. Simple genetic distance calculations are then used to undercount the number of mutations that have occurred in evolutionary history.

There are various types of substitution models that could have been chosen for analyses. The Kimura 2-parameter model was used with the ML analyses. I chose this model because it allows for differing rates of transitions and tranversions. Jukes-Cantor model does not take into account that transitions rates (between purines) A↔G and (between pyrmidine) C↔T are

different from transversions rates of A↔C, A↔T, C↔G, G↔T. I wanted to make sure that even these transitions and transversions were covered in the phylogenetic tree construction.

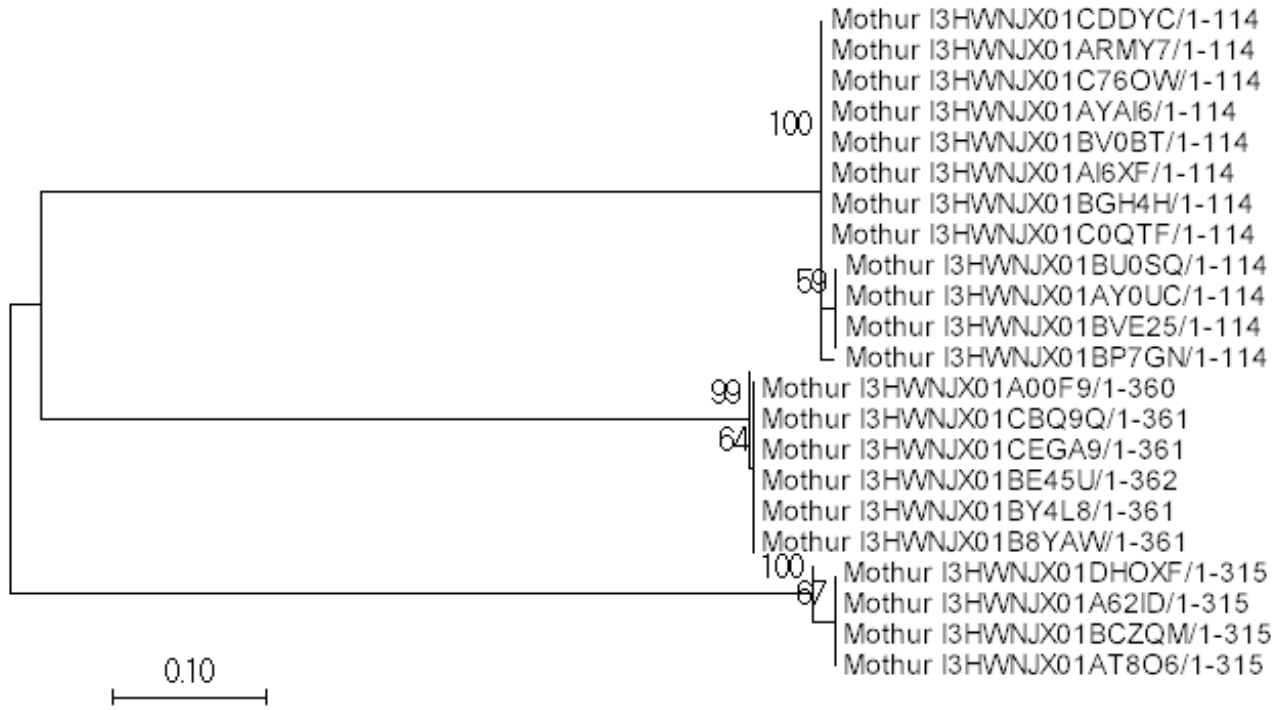


Figure 3.9: The phylogenetic tree constructed for the Honeybush natural sample. The tree was constructed by using maximum likelihood with MEGA7.

The evolutionary history was inferred by using the Maximum Likelihood method based on the Kimura 2-parameter model (Kimura, 1980). The tree with the highest log likelihood (-1064.8114) was shown. The percentage of trees in which the associated taxa clustered together was shown next to the branches. Initial tree(s) for the heuristic search were obtained by applying the Neighbor-Joining method to a matrix of pairwise distances estimated using the Maximum Composite Likelihood (MCL) approach. A discrete Gamma distribution was used to model evolutionary rate differences among sites (5 categories (+G, parameter = 200.0000)). The tree was drawn to scale, with branch lengths measured in the number of substitutions per site. The analysis involved 64 nucleotide sequences. There were a total of 412 positions in the final dataset. Evolutionary analyses were conducted in MEGA7 (Kumar *et al*., 2015).

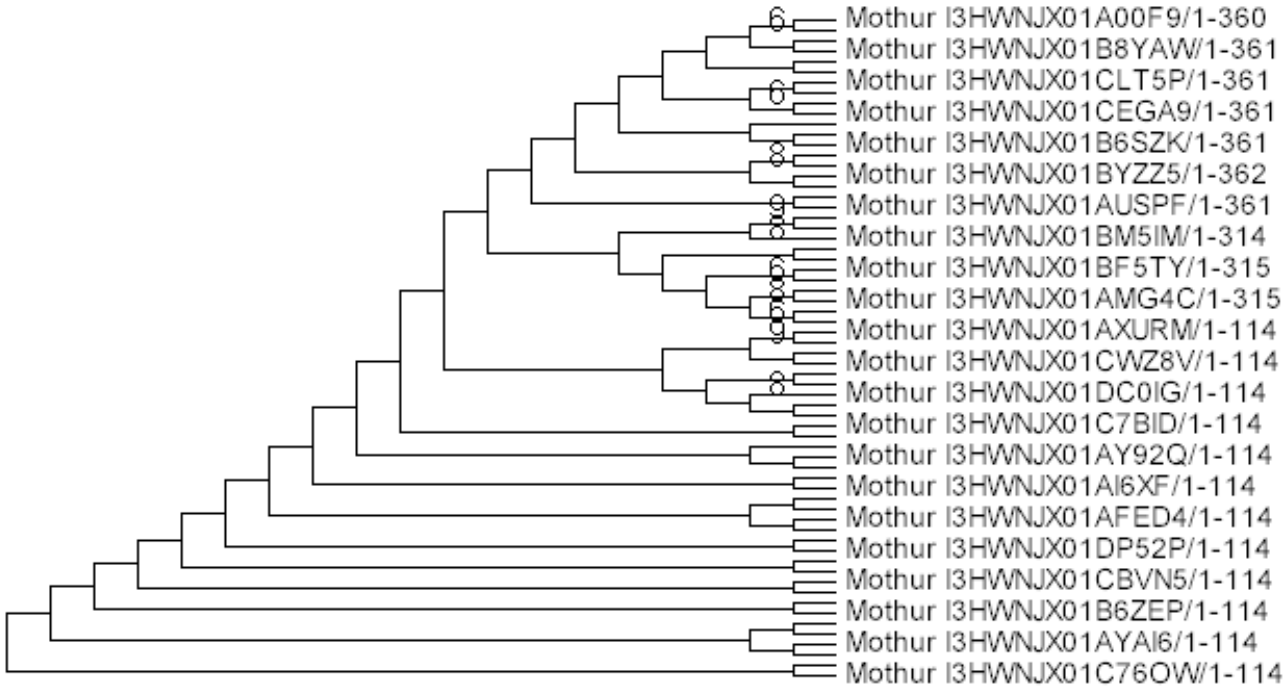


Figure 3.10: The phylogenetic tree calculated for the Honeybush natural sample using bootstrapping. This tree was also constructed using maximum likelihood in MEGA7.

The evolutionary history was inferred by using the Maximum Likelihood method based on the Kimura 2-parameter model. The bootstrap consensus tree inferred from 500 replicates (Felsenstein, 1985) was taken to represent the evolutionary history of the taxa analyzed (Felsenstein, 1985). Branches corresponding to partitions reproduced in less than 50% bootstrap replicates were collapsed. The percentage of replicate trees in which the associated taxa clustered together in the bootstrap test (500 replicates) were shown next to the branches [3]. Initial tree(s) for the heuristic search were obtained by applying the Neighbor-Joining method to a matrix of pairwise distances estimated using the Maximum Composite Likelihood (MCL) approach. A discrete Gamma distribution was used to model evolutionary rate differences among sites (5 categories (+G, parameter = 200.0000)). The analysis involved 64 nucleotide sequences. There were a total of 412 positions in the final dataset. Evolutionary analyses were conducted in MEGA7.

Figure 3.10 can be seen as a circle tree in figure 3.11 where the percentage of replicate trees, in which the associated taxa clustered together in the bootstrap test, can be seen more clearly.

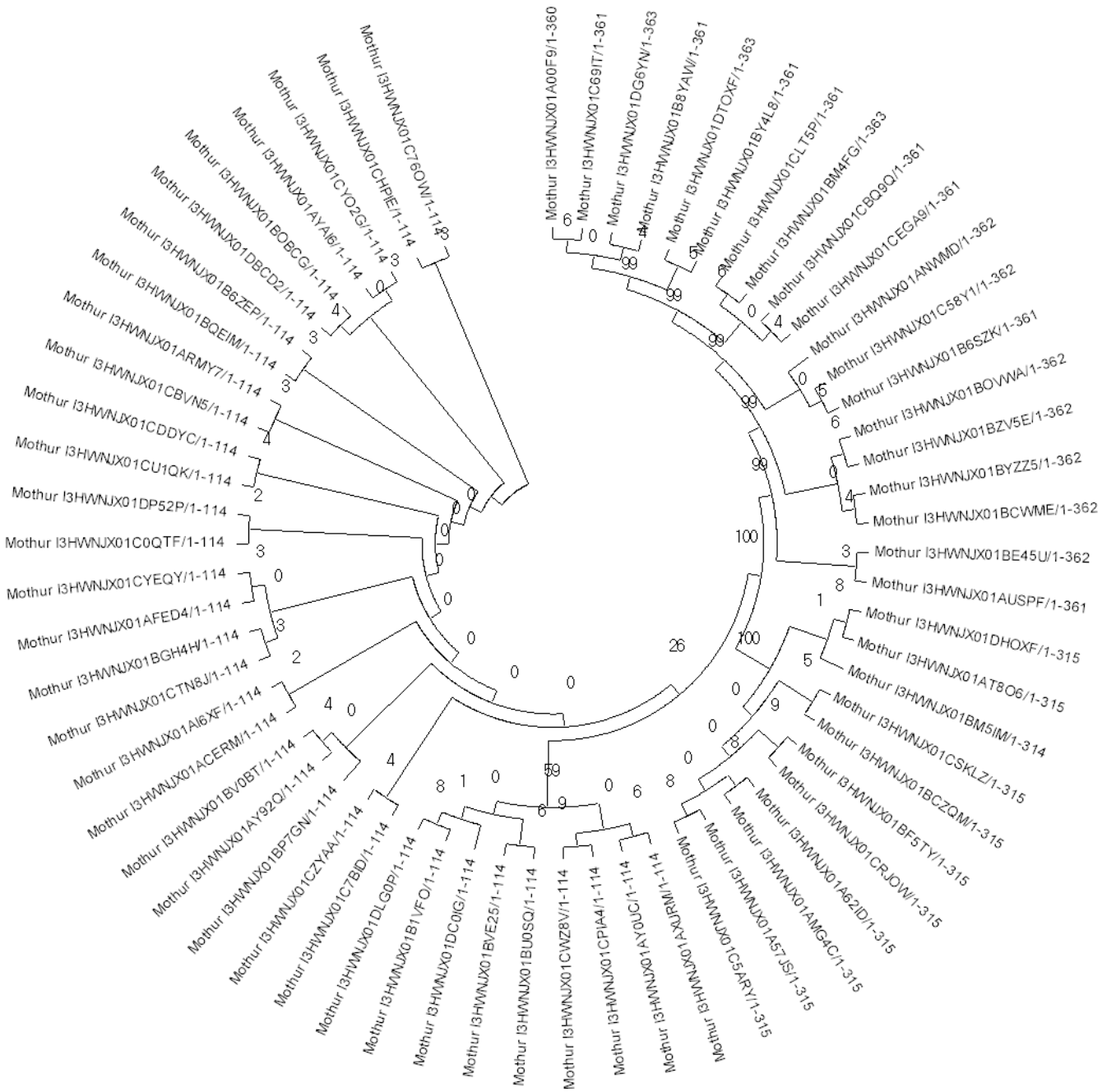


Figure 3.11: The circle phylogenetic tree of the Honeybush natural sample using bootstrapping. This tree was constructed in MEGA7 using maximum likelihood.

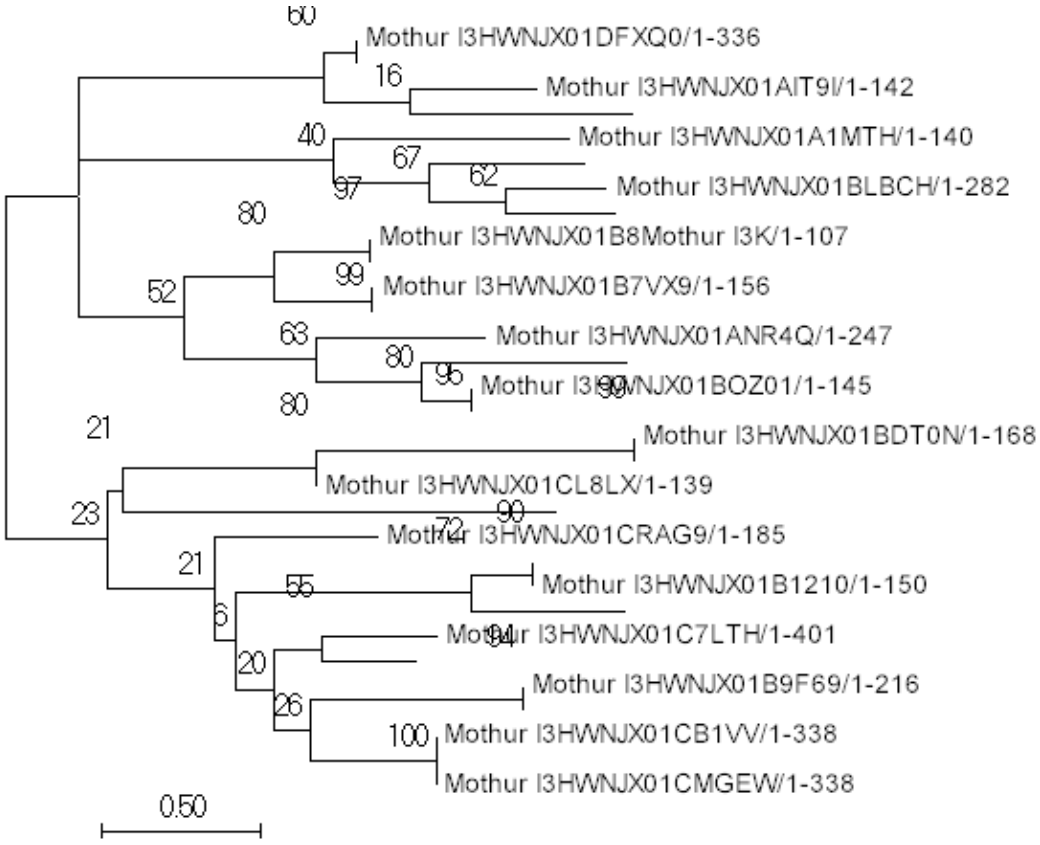


Figure 3.12: The phylogenetic tree constructed for the Honeybush commercial sample. This tree was constructed using maximum likelihood with the program MEGA7.

The evolutionary history was inferred by using the Maximum Likelihood method based on the Kimura 2-parameter model. The tree with the highest log likelihood (-4485.7858) was shown. Initial tree(s) for the heuristic search were obtained by applying the Neighbor-Joining method to a matrix of pairwise distances estimated using the Maximum Composite Likelihood (MCL) approach. A discrete Gamma distribution was used to model evolutionary rate differences among sites (5 categories (+G, parameter = 18.5256)). The tree was drawn to scale, with branch lengths measured in the number of substitutions per site. The analysis involved 64 nucleotide sequences. There were a total of 507 positions in the final dataset. Evolutionary analyses were conducted in MEGA7.

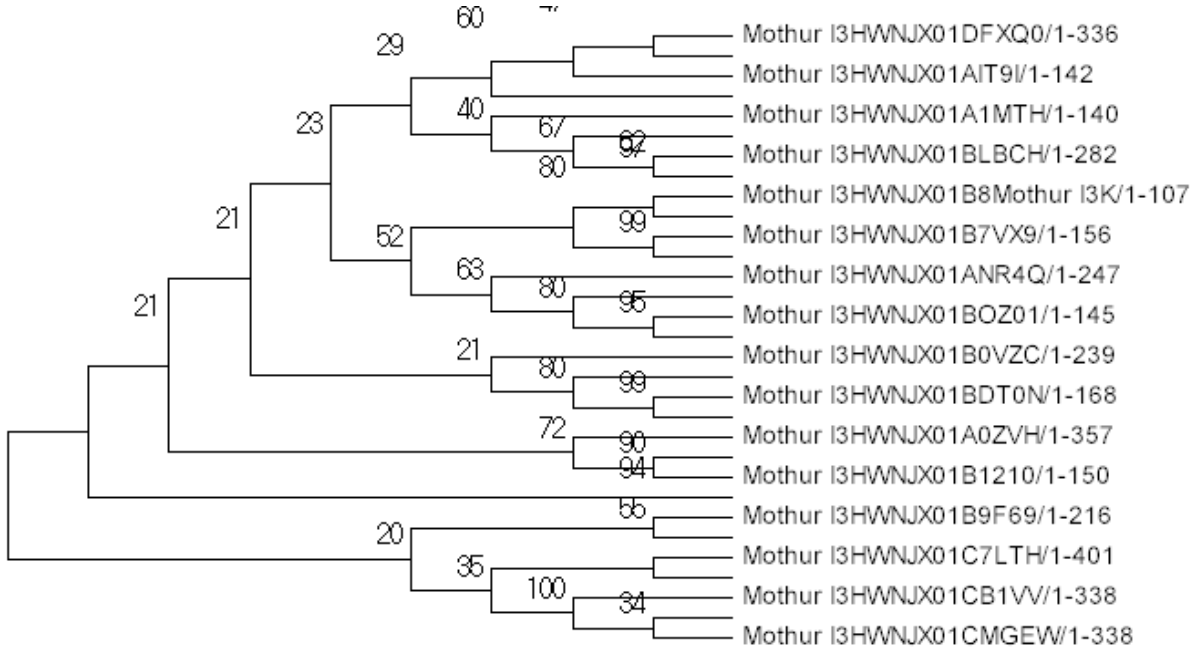


Figure 3.13: The phylogenetic tree constructed for the Honeybush commercial sample. This tree was constructed by using bootstrapping with the maximum likelihood method with MEGA7.

The evolutionary history was inferred by using the Maximum Likelihood method based on the Kimura 2-parameter model. The bootstrap consensus tree inferred from 500 replicates was taken to represent the evolutionary history of the taxa analyzed. Branches corresponding to partitions reproduced in less than 50% bootstrap replicates were collapsed. Initial tree(s) for the heuristic search were obtained by applying the Neighbor-Joining method to a matrix of pairwise distances estimated using the Maximum Composite Likelihood (MCL) approach. A discrete Gamma distribution was used to model evolutionary rate differences among sites (5 categories (+G, parameter = 18.5256)). The analysis involved 64 nucleotide sequences. There were a total of 507 positions in the final dataset. Evolutionary analyses were conducted in MEGA7.

Figure 3.13 can be seen as a circle tree in figure 3.14 where the percentage of replicate trees, in which the associated taxa clustered together in the bootstrap test, can be seen more clearly.

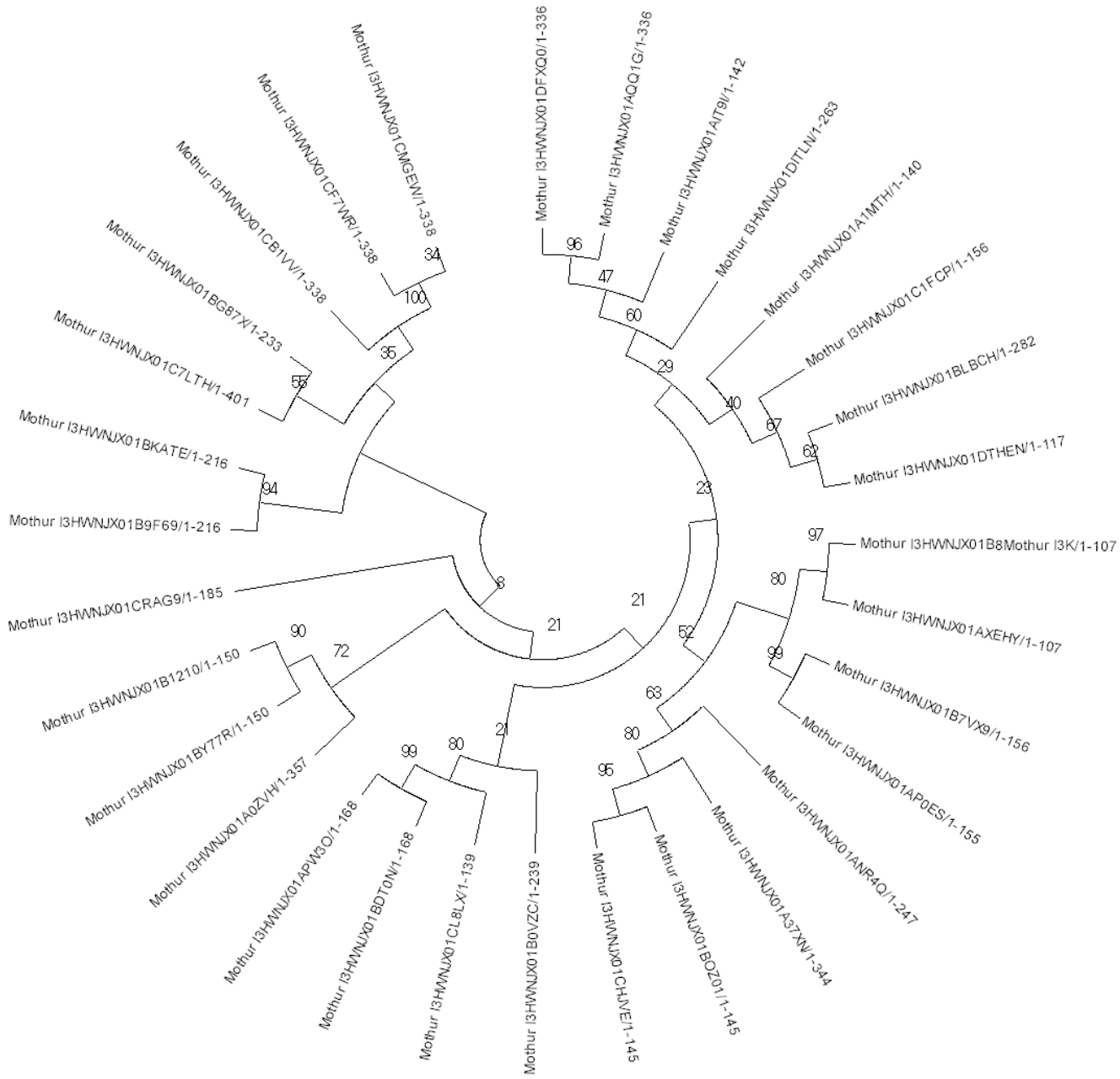


Figure 3.14: The phylogenetic tree constructed for the Honeybush commercial sample. The tree was constructed by using bootstrapping and the maximum likelihood method with MEGA7.

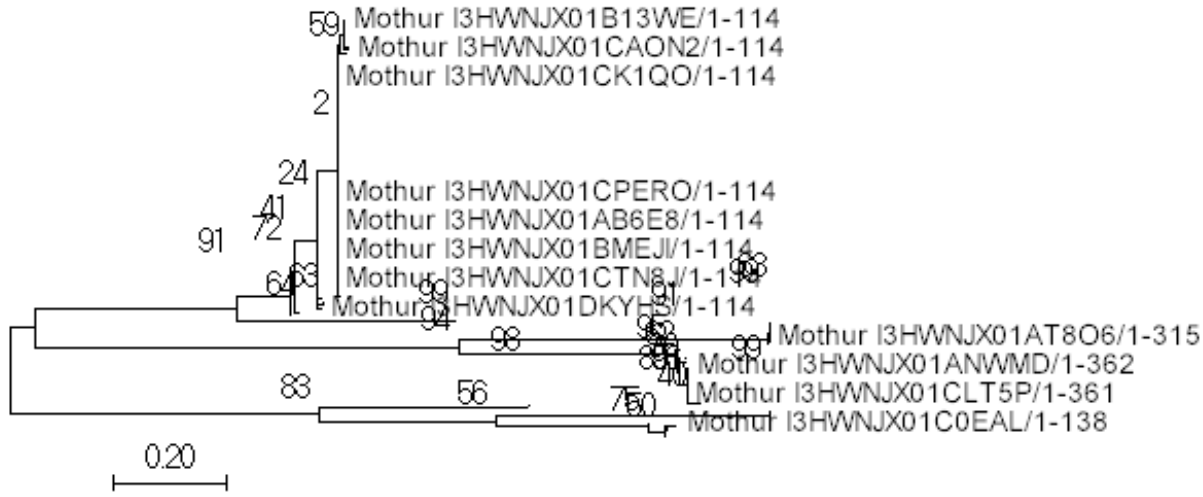


Figure 3.15: The phylogenetic tree constructed for the Rooibos natural sample. The tree was constructed by using the maximum likelihood method with MEGA7.

The evolutionary history was inferred by using the Maximum Likelihood method based on the Kimura 2-parameter model. The tree with the highest log likelihood (-2318.5651) was shown. Initial tree(s) for the heuristic search were obtained by applying the Neighbor-Joining method to a matrix of pairwise distances estimated using the Maximum Composite Likelihood (MCL) approach. A discrete Gamma distribution was used to model evolutionary rate differences among sites (5 categories (+G, parameter = 163.3969)). The tree was drawn to scale, with branch lengths measured in the number of substitutions per site. The analysis involved 204 nucleotide sequences. There were a total of 419 positions in the final dataset. Evolutionary analyses were conducted in MEGA7.

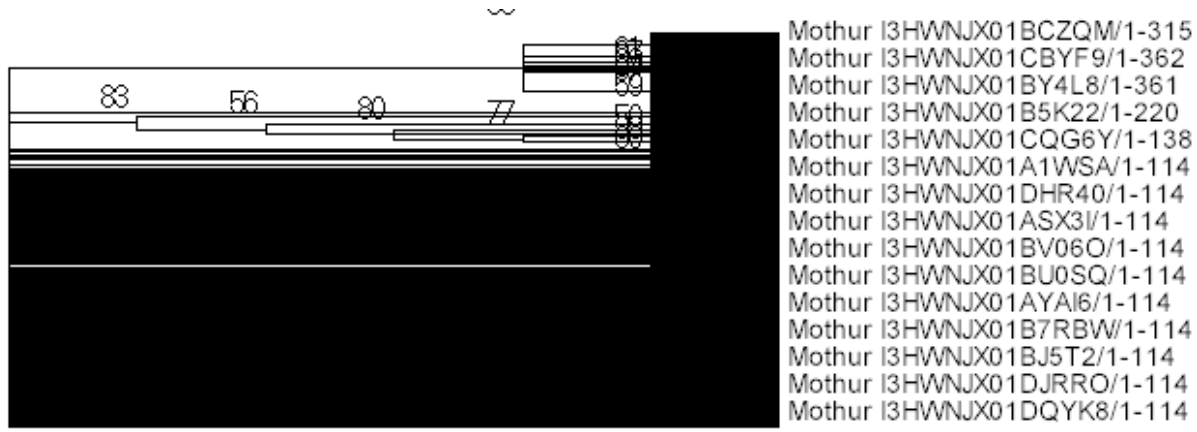


Figure 3.16: The phylogenetic tree constructed for the Rooibos natural sample. This tree was constructed by using bootstrapping with maximum likelihood with MEGA7.

The evolutionary history was inferred by using the Maximum Likelihood method based on the Kimura 2-parameter model. The bootstrap consensus tree inferred from 500 replicates was taken to represent the evolutionary history of the taxa analyzed. Branches corresponding to partitions reproduced in less than 50% bootstrap replicates were collapsed. Initial tree(s) for the heuristic search were obtained by applying the Neighbor-Joining method to a matrix of pairwise distances estimated using the Maximum Composite Likelihood (MCL) approach. A discrete Gamma distribution was used to model evolutionary rate differences among sites (5 categories (+G, parameter = 163.3969)). The analysis involved 204 nucleotide sequences. There were a total of 419 positions in the final dataset. Evolutionary analyses were conducted in MEGA7.

Figure 3.16 can be seen as a circle tree in figure 3.17 where the percentage of replicate trees, in which the associated taxa clustered together in the bootstrap test, can be seen more clearly.

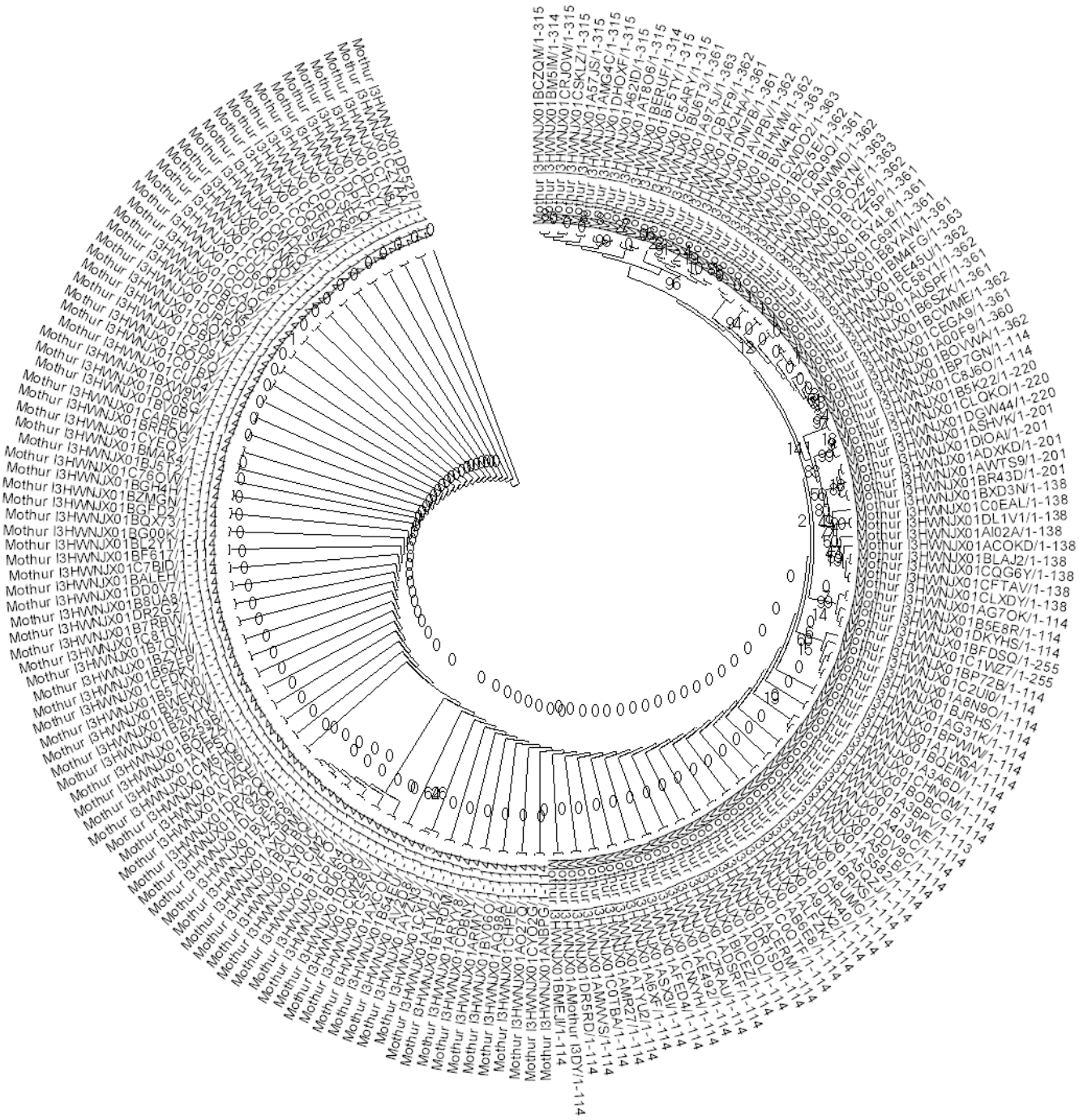


Figure 3.17: The circle phylogenetic tree constructed for the Rooibos natural sample. The tree was constructed by using the maximum likelihood method and bootstrapping with MEGA7.

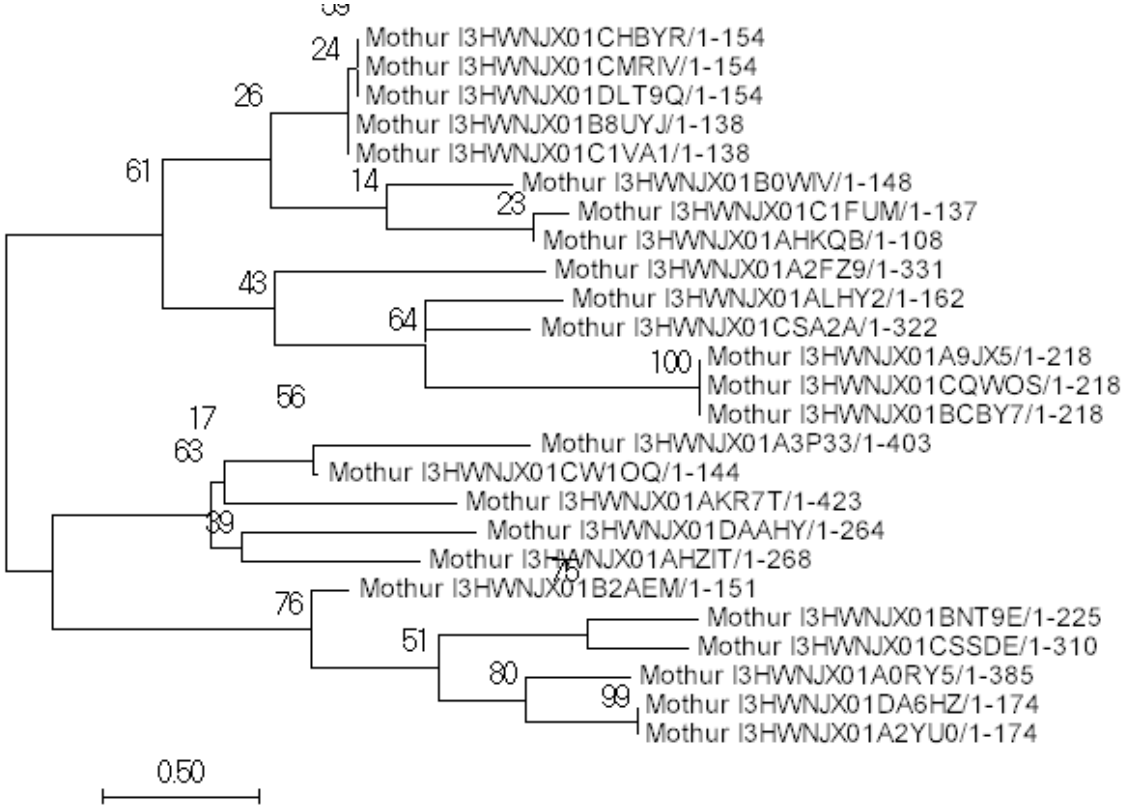


Figure 3.18: The phylogenetic tree calculated for the Rooibos commercial sample. The tree was constructed by using maximum likelihood with MEGA7.

The evolutionary history was inferred by using the Maximum Likelihood method based on the Kimura 2-parameter model. The tree with the highest log likelihood (-5747.9695) was shown. Initial tree(s) for the heuristic search were obtained by applying the Neighbor-Joining method to a matrix of pairwise distances estimated using the Maximum Composite Likelihood (MCL) approach. A discrete Gamma distribution was used to model evolutionary rate differences among sites (5 categories (+G, parameter = 7.6918)). The tree was drawn to scale, with branch lengths measured in the number of substitutions per site. The analysis involved 25 nucleotide sequences. There were a total of 585 positions in the final dataset. Evolutionary analyses were conducted in MEGA7.

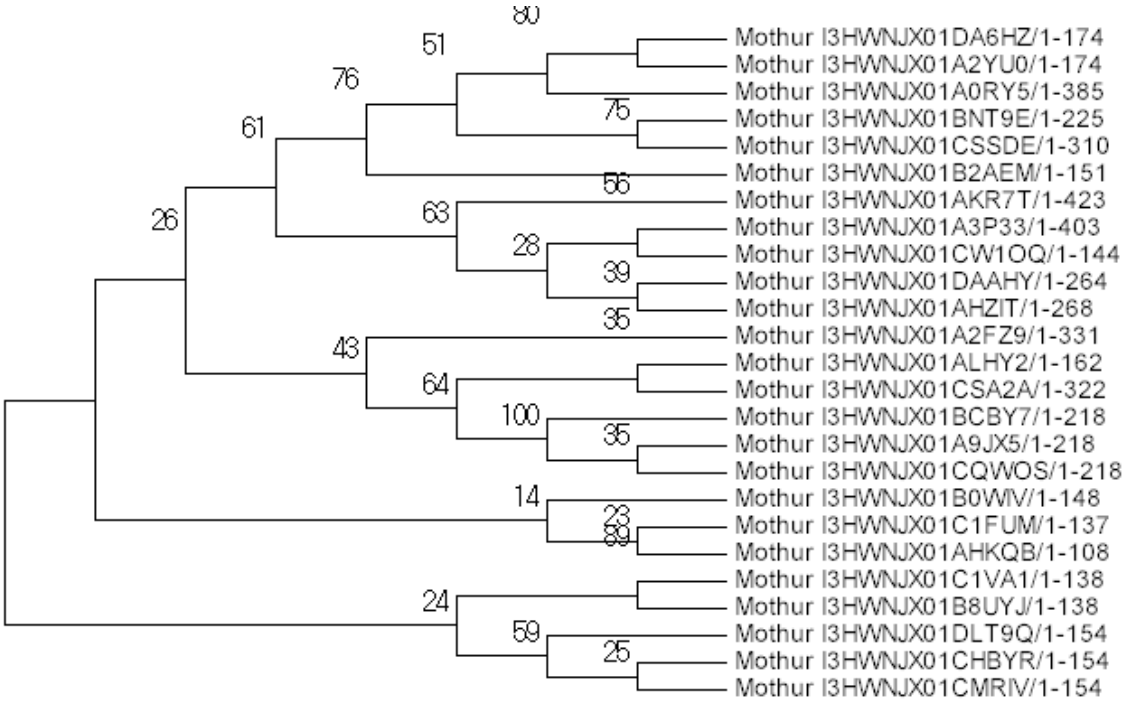


Figure 3.19: The phylogenetic tree constructed for the Rooibos commercial sample. The tree was constructed by using bootstrapping and maximum likelihood with MEGA7.

The evolutionary history was inferred by using the Maximum Likelihood method based on the Kimura 2-parameter model. The bootstrap consensus tree inferred from 500 replicates was taken to represent the evolutionary history of the taxa analyzed. Branches corresponding to partitions reproduced in less than 50% bootstrap replicates were collapsed. Initial tree(s) for the heuristic search were obtained by applying the Neighbor-Joining method to a matrix of pairwise distances estimated using the Maximum Composite Likelihood (MCL) approach. A discrete Gamma distribution was used to model evolutionary rate differences among sites (5 categories (+G, parameter = 7.6918)). The analysis involved 25 nucleotide sequences. There were a total of 585 positions in the final dataset. Evolutionary analyses were conducted in MEGA7.

Figure 3.19 can be seen as a circle tree in figure 3.20 where the percentage of replicate trees, in which the associated taxa clustered together in the bootstrap test, can be seen more clearly.

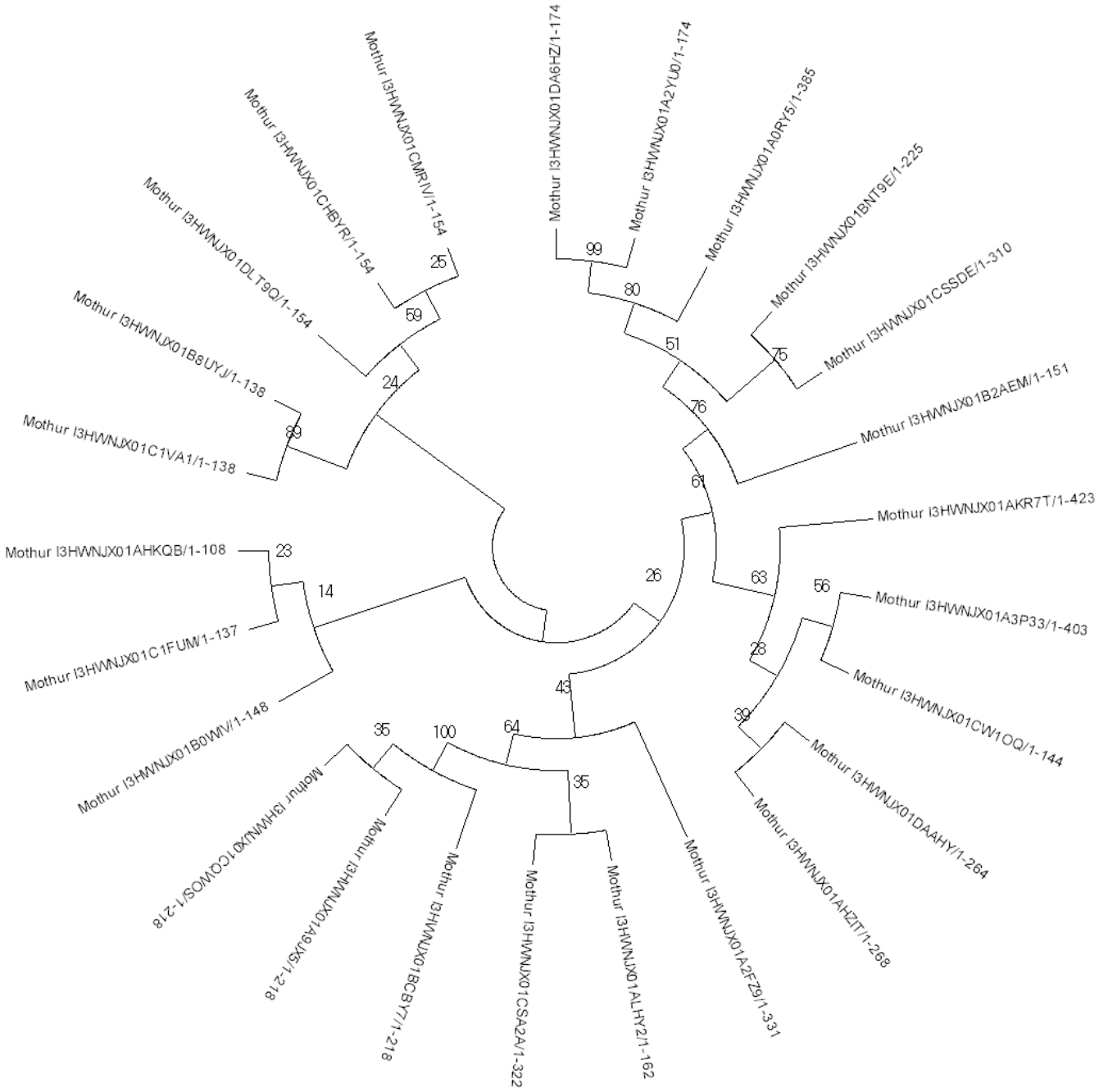


Figure 3.20: The phylogentic tree calculated for the Rooibos commercial sample. The tree was calculated by using bootstrapping, and maximum likelihood with MEGA7.

3.4.5) Optional methods

The methodology and results regarding the optional, standalone programs also studied, are in appendices III, IV, V, and VII. The discussions of the results of these methods can also be found in the named appendices.

Conclusion

This project was a study about the different available pipelines currently available to analyse 18S AM fungal data. It is important to take note that the three main aims of this study was to perform the quality control and initial analysis of raw DNA reads that were generated by Roche 454 technology from AM communities of natural and commercial Rooibos and Honeybush plants; to identify the analytical pipeline that suited best to work with the Roche 454 generated AM metagenomic data; and to identify the fungal species and OTU's present in the sequenced metagenomic community. The tools, pipelines and databases studied during this project can be used to design a suitable pipeline that will be able to analyse 18S AM fungal data with ease.

The pipelines focused on during this study were QIIME, Mothur and RDP, which were also compared. The databases focused on were Silva 104 and MaarjAM. Various obstacles were encountered during this study regarding these pipelines and databases.

There were various challenges that were important to remember when doing metagenomic analysis: firstly, there were problems with the assembly of the reads as the total number of the organisms within the samples were not known beforehand and the read coverage was not sufficient in order to uncover all the organisms within the samples. This will allways be a problem with organisms that cannot be cultured. Secondly, most of the organisms in the samples have not been sequenced before and thus there were no homologs available, or sequencing data in databases.

Sufficient QC studies (Chapter 2) were done with QIIME, Mothur and the RDP pipelines. What was found was that the results from the QIIME and Mothur pipelines were similar to each other. The reason for this had to be that both QIIME and Mothur do have parts in their pipelines, and especially the QC steps (as in table I below), that were able to work with the AM fungal sequences. The RDP pipeline, on the other hand, did not have any reference databases, nor reference genes that could be used when working with the 18S AM fungal data.

Table I: Summary of the QC results found using the three different pipelines. This table shows a comparison between the mean values of each important factor when doing QC. The

number of sequences is not the mean value, but rather the maximum amount of sequences present within the samples.

Pipeline	Sample	Before/After QC	Start	End	NBases	Ambigs	Polymer	NumSeqs
QIIME	Honeybush natural	Before	1	289.003	289.003	0.0141081	4.43481	17295
		After	0.908228	151.927	151.973	0.000791139	3.66614	1264
	Honeybush commercial	Before	1	472.099	472.099	0.0155298	5.34667	10174
		After	1	134.939	134.939	0	3.56463	147
	Rooibos natural	Before	1	398.305	398.305	0.0153352	4.99345	16498
		After	0.996581	189.162	189.162	0	3.95556	585
	Rooibos commercial	Before	1	459.287	459.287	0.0178084	5.43385	4437
		After	1	152.366	152.366	0	3.56098	41
Mothur	Honeybush natural	Before	1	289.003	289.003	0.0141081	4.43481	17295
		After	0.908228	151.927	151.973	0.000791139	3.66614	1264
	Honeybush commercial	Before	1	427.099	427.099	0.0155298	5.34667	10174
		After	0.840491	121.638	121.718	0	3.31288	163
	Rooibos natural	Before	1	398.305	398.305	0.0153352	4.99345	16498
		After	0.927393	182.573	182.609	0	3.85314	606
	Rooibos commercial	Before	1	459.287	459.287	0.0178048	5.43385	4437
		After	0.863636	141.909	141.977	0	3.38636	44
RDP	Honeybush natural	Before	0.993293	279.322	279.326	0.0141081	4.41208	17295
	Honeybush commercial	Before	0.998231	461.037	461.038	0.0155298	5.33586	10174
	Rooibos natural	Before	1	390.589	390.589	0.0153352	4.99212	16498
	Rooibos commercial	Before	1	445.725	445.725	0.0178048	5.43047	4437

From the QC results comparisons above, it can be seen that the QIIME and Mothur pipelines do give roughly similar results. The QC results of the RDP pipeline could not be generated seeing that there was no reference gene that could be used, but even if it could be generated,

the first step of the pipeline was done against the fungal ITS gene, and this makes it unreasonable to use as a pipeline. It was thought best to rather look at the QIIME and Mothur pipelines in future until there is an adequate reference gene available on the RDP pipeline. The RDP pipeline was still used in further analysis (where possible) in chapter 3, but for future analysis it would be wise to not consider the RDP pipeline.

The one major problem encountered when doing OTU picking (chapter 3) was that the sequences present in the reference databases were not of much use. The databases used as references were Silva 104 and MaarjAM. It is not clear yet as to why the MaarjAM database did not give good results when using it in the Mothur pipeline as the reference, seeing as this database contained AM fungi sequences as virtual taxa. The Silva 104 database did not give good results when using it with the QIIME pipeline, but this was understandable seeing that this database contained very little AM fungal sequences.

Classification was able to be done with the RDP pipeline (chapter 3). Even though this classification was done in the RDP pipeline where no sufficient reference gene was provided, the classification was done against the ITS reference gene. The results from this showed that all the sequences present in the samples were from fungal origin, but as the classification progressed downward, the amount of sequences that were able to be given a classification, decreased by each classification level.

From the analyses of the sequences done in this study, it is recommended to use either the QIIME or Mothur pipelines to do QC and initial processing of the sequence reads. The main reason for this is that there are more resources that use fungi genes as reference than in the RDP pipeline. It is possible to use the unsupervised workflow in RDP, but in order to do that, a database of AM fungi need to be made (this recommendation is discussed more below). At this stage, statistical analysis would also be more accurate with the QIIME and Mothur pipelines, seeing that the RDP pipeline statistical analysis has been based on that of 16S rRNA and not 18S rRNA. Again, referring back to an RDP Staff member stating that not much effort is going into setting up an 18S rRNA reference dataset.

When doing the multiple sequence alignments (Appendix V, chapter 3) it was clear that when shorter sequences were left in the files to be aligned, the alignments were done insufficiently and no phylogenetic trees could be constructed. This was due to the neighbor joining method of constructing the phylogenetic trees only doing tree construction up to the last base of the

shortest read within the file. Due to this, all sequences shorter than 100 bp were taken out of the files and alignments and tree constructions were done. With the neighbor joining trees with Jalview, all the samples, both natural and commercial, were found to have two major branches in their phylogenetic trees and the diversity in the natural samples were found to be much more than that in the commercial samples.

The evolutionary history was inferred by using the Maximum Likelihood method based on the Kimura 2-parameter model. Initial trees for the heuristic search were obtained by applying the Neighbor-Joining method to a matrix of pairwise distances estimated using the Maximum Composite Likelihood approach. The highest log likelihood trees were shown in the original trees where a discrete Gamma distribution was used to model evolutionary rate differences among sites. The trees were drawn to scale, with branch lengths measured in the number of substitutions per site. The number of positions in the final datasets were more in the natural samples that that in the commercial samples. In the bootstrap consensus trees, the evolutionary history was inferred and the percentage of replicate trees in which the associated taxa were clustered together were shown next to the branches. A discrete Gamma distribution was used to model evolutionary rate differences among sites.

The expected outcome was to find that there were a higher diversity and abundance of fungi in the natural than in the commercial samples. This could have been a result of the unnatural growth environments of the Rooibos and Honeybush plantations, and also due to the use of pesticides and not purposefully trying to maintain the AM fungi cultures within these environments. The outcome of this study lead to the realization that there are still not enough sequencing data of AM fungi available in the present databases. This lead to difficulties designing a pipeline and analyzing the sequencing data produced by Rhodes University. Further studies will need to be done in order to find a suitable pipeline, or even design a suitable pipeline, to be able to effectively analyse 18S AM fungal data.

From this study it became clear that there is still a lot of work that needs to be done in order to be able to design a sufficient pipeline to analyse 18S AM fungal data. This will become easier when there are reference databases available that contain sufficient amounts of AM fungal data from around the world. It is necessary to be open-minded when working on the 18S AM fungal data in order to find new and creative ways to deal with this type of data and these types of

problems. What might be needed is to integrate the QIIME and Mothur pipelines in future, and also more than one reference database, in order to achieve the most accurate results.

Another recommendation may be to consider using a locally built database that can be developed and grown as more sequence data become available. The sequencing data can also include additional steps as cloning and full sequencing. An existing pipeline or a newly built one, can then be adapted to work with this locally built database. Also, the taxa that do not blast to anything can then be assigned a code and metadata will then show an affinity to a particular coded group. This data can then be used in downstream statistical analysis.

References

Abbott, J.C., Aanensen, D.M., Rutherford, K., Butcher, S. and Spratt, B.G. (2005) WebACT—an online

companion for the Artemis Comparison Tool, *Bioinformatics*, 21(18): 3665-3666.

Abreu-Goodger, C., & Merino, E. (2005) RibEx: a web server for locating riboswitches and other conserved bacterial regulatory elements, *Nucleic acids research*, *33*(suppl 2): W690-W692.

Alonso-Alemany, D., Clemente, J. and Jansson, J. (2011) Taxonomic assignment in metagenomics with TANGO, *EM Bnet Journal,* 17: 46-50.

Angelard, C., Colard, A., Niculita-Hirzel, H., Croll, D. and Sanders, I.R. (2010) Segregation in a mycorrhizal fungus alters rice growth and symbiosis-specific gene transcription, *Curr Biol*, 20: 1216-1221.

Ashelfold, K.E., Chuzhanova, N.A. and Fry, J.C. (2005) At least 1 in 20 16S rRNA sequence record currently held in public repositories is estimated to contain substantial anomalies, *Appl Environ Microbiol*, 71: 7724-7736.

Aziz, R.K., Bartels, D., Best, A.A., DeJongh, M., Disz, T., Edwards, R.A., ... and Meyer, F. (2008) The RAST Server: rapid annotations using subsystems technology, *BMC genomics*, *9*(1): 75.

Badri, D.V., Zolla, G., Bakker, M.G., Manter, D.K. and Vivanco, J.M. (2013) Potential impact of soil microbiomes on the leaf metabolome and on herbivore feeding behavior, *New Phytologist*, *198*(1): 264-273.

Blackwell, M. (2011) The Fungi 1, 2, 3 ... 5.1 million species?, *American Journal of Botany*, 98(3): 426-438.

Bolan, N.S. (1991) A critical review on the role of mycorrhizal fungi in the uptake of phosphorus by plants, *Plant Soil,* 134: 189-207.

Caporaso, J.G., Kuczynski, J., Stombaugh, J., Bittinger, K., Bushman, F.D., Costello, E.K., Fierer, N., Peña, A.G., Goodrich, J.K., Gordon, J.I., Huttley, G.A., Kelley, S.T., Knights, D., Koenig, J.E., Ley, R.E., Lozupone, C.A., McDonald, D., Muegge, M.D., Pirrung, M., Reeder, J., Sevinsky, J.R., Turnbaugh, P.J., Walters, W.A., Widmann, J., Yatsunenko, T., Zaneveld, J. and Knight, R. (2010) QIIME allows analysis of high-throughput community sequencing data, *Nat Methods*, 7(5):335-336.

Carver, T.J., Rutherford, K.M., Berriman, M., Rajandream, M., Barrell, B.G. and Parkhill, J. (2005) ACT: the Artemis comparison tool, *Bioinformatics*, 21(16): 3422-3423.

Carver, T., Thomson, N., Bleasby, A., Berriman, M., and Parkhill, J. (2008) DNAPlotter: circular and linear interactive genome visualization, *Bioinformatics*, 25(1): 119-120.

Chan, M., Ji, S.M., Yeo, Z.X., Gan, L., Yap, E., Yap, Y.S., Ng, R., Tan, P.H., Ho, G.H., Ang, P. and Lee, A.S.G. (2012) Development of a Next-Generation Sequencing Method for *BRCA* Mutation Screening: A Comparison between a High-Throughput and a Benchtop Platform, *The Journal of Molecular Diagnostics*, 14(6): 602-612.

Chowdhury, F. and Chowdhury, A. (2012) Pyrosequencing – An Alternative to Traditional Sanger Sequencing, *American Journal of Biochemistry and Biotechnology*, 8(1): 14-20.

Chun, J., Lee, J.H. and Jung, Y. (2007) EzTaxon: a web-based tool for the identification of prokaryotes based on 16S ribosomal RNA gene sequences, *Int J Syst Microbiol*, 57: 2259-2261.

Clark, R.B. and Zeto, S.K. (2000) Mineral acquisition by arbuscular mycorrhizal plants, *J. Plant Nutr,* 23: 867-902.

Cole, J.R., Wang, Q., Cardenas, E., Fish, J., Chai, R., Farris, R.J., Kulam-Syed-Mohideen, A.S., McGarrell, D.M., Marsh, T., Garrity, G.M. and Tiedje, J.M. (2009) The Ribosomal Database Project: improved alignments and new tools for rRNA analysis, *Nucleic Acids Res,* 37(Database issue): D141-145; doi: 10.1093/nar/gkn879 [PMID: 19004872].

Cole, J.R., Wang, Q., Fish, J.A., Chai, B., McGarrell, D.M., Sun, Y., Brown, C.T., Porras-Alfaro, A., Kuske, C.R. and Tiedje, J.M. (2014) Ribosomal Database Project: data and tools for high throughput rRNA analysis, *Nucleic Acids Res*, 42 (Database issue): D633 - D642.

Cole, J.R., Wang, Q., Fish, J.A., Chai, B., McGarrell, D.M., Sun, Y., Brown, C.T., Porras-Alfaro, A., Kuske, C.R. and Tiedje, J.M. (2013) Ribosomal Database Project: data and tools for high throughput rRNA analysis, *Nucleic Acids Res*, 42 (Database issue): doi: 10.1093/nar/gkt1244**.**

Cole, J.R., Wang, Q., Chai, B. and Tiedje, J.M. (2011) The Ribosomal Database Project: sequences and software for high-throughput rRNA analysis. In F.J. de Bruijn (ed), Handbook of Molecular Microbial Ecology I: Metagenomics and Complementary Approaches, J. Wiley & Sons, Inc., Hoboken, NJ.

Costini, M.L., Calizza, E. and Rossi, L. (2014) Stable isotope variation during fungal colonisation of leaf detritus in aquatic environments, *Fungal Ecology*, 11: 154-163.

Derrien, T., André, C., Galibert, F. and Hitte, C. (2007) AutoGRAPH: an interactive web server for automating and visualizing comparative genome maps, *Bioinformatics*, *23*(4): 498-499.

Dickson, S. (2004) The *Arum-Paris* continuum of mycorrhizal symbioses, *New Phytol*, 163: 187-200.

Edgar, R.C. (2010) Search and clustering orders of magnitude faster than BLAST, *Bioinformatics*, 26: 2460-2461.

Edgar, R.C. (2013) UPARSE: Highly accurate OTU sequences from microbial amplicon reads, *Nature Methods*, 10, 996–998 (2013) doi:10.1038/nmeth.2604.

Edgar, R.C., Haas, B.J., Clemente, J.C., Quince, C. and Knight, R. (2011) UCHIME improves sensitivity and speed of chimera detection, *Bioinformatics,* doi: 10.1093/bioinformatics/btr381 [PMID 21700674].

Felsenstein, J. (1985) Confidence limits on phylogenies: An approach using the bootstrap, *Evolution,* 39:783-791.

Fish, J.A., Chai, B., Wang, Q., Sun, Y., Brown, C.T., Tiedje, J.M. and Cole, J.R. (2013) FunGene: the Functional Gene Pipeline and Repository, *Front. Microbiol,* 4:291; doi: 10.3389/fmicb.2013.00291

Gallotte, A., Van Tuinen, D. and Atkinson, D. (2004) Diversity of arbuscular mycorrhizal fungi colonising roots of the grass species *Agrostis capillaris* and *Lolium perenne* in a field experiment, *Mycorrhiza*, 14: 111-117.

Genre, A., Chabaud, M., Faccio, A., Barker, D.G. and Bonfante, P. (2008) Prepenetration apparatus assembly precedes and predicts the colonization patterns of arbuscular mycorrhizal fungi within the root cortex of both *Medicago truncatula* and *Daucus carota*, *Plant Cell*, 20: 1407-1420.

Gilbert, J.A., Meyer, F., Jansson, J., Gordon, J., Pace, N., Tiedje, J. and Glöckner, F.O. (2010) The Earth Microbiome Project: Meeting report of the "1st EMP meeting on sample selection and acquisition" at Argonne National Laboratory October 6th 2010, *Standards in genomic sciences*, 3(3): 249.

Gîrdea, M., Noé, L., and Kucherov, G. (2010) Back-translation for discovering distant protein homologies in the presence of frameshift mutations, *Algorithms for Molecular Biology*, 5(6).

Goloboff, P.A. (2003) Parsimony, likelihood, and simplicity, *Cladistics*, 19(2): 91-103.

Gómez, J.M., Verdú, M. and Perfectti, F. (2010) Ecological interactions are evolutionarily conserved across the entire tree of life, *Nature*, 465: 918-921.

Goris, J., Konstantinidis, K.T., Klappenbach, J.A., Coenye, T., Vandamme, P. and Tiedje, J.M. (2007) DNA-DNA hybridization values and their relationship to whole-genome sequence similarities, *Int J Syst Evol Microbiol,* 57(Pt 1): 81-91.

Grant, J.R., and Stothard, P. (2008) The CGView Server: a comparative genomics tool for circular genomes, *Nucleic acids research*, *36*(suppl 2): W181-W184.

Hamady, M., Walker, J.J., Harris, J.K., Gold, N.J., and Knight, R. (2008) Error-correcting barcoded primers allow hundreds of samples to be pyrosequenced in multiplex, *Nature Methods*, 5(3): 235–237.

Hasman, H., Saputra, D., Sicheritz-Ponten, T., Lund, O., Svendsen, C.A., Frimodt-Møller, N. and Aarestrup, F.M. (2013) Rapid whole genome sequencing for the detection and characterization of microorganisms directly from clinical samples *Journal of clinical microbiology*, JCM-02452.

Hawksworth, D.L. (1991) The magnitude of fungal diversity: the 1.5 million species estimate revisited, *Mycological Research*, 105: 1422-1432.

Hijri, M. and Sanders, I.R. (2005) Low gene copy number shows that arbuscular mycorrhizal fungi inherit genetically different nuclei, *Nature*, 433: 160-163.

Holt, C., and Yandell, M. (2011) MAKER2: an annotation pipeline and genome-database management tool for

second-generation genome projects, *BMC bioinformatics*, *12*(1): 491.

Hopkin, M. (2006) Ecology: spying on nature, *Nature*, 444(7118): 420-421.

Hu, B., Jin, J., Guo, A.Y., Zhang, H., Luo, J., and Gao, G. (2014) GSDS 2.0: an upgraded gene feature visualization server, *Bioinformatics*, btu817.

Huang, X., and Madan, A. (1999) CAP3: A DNA sequence assembly program, *Genome research*, 9(9): 868-877.

Husan, D.H., Auch, A.F., Qi, J. and Schuster, S.C. (2007) MEGAN analysis of metagenomic data, *Genome Res*, 17(3): 377-386.

Husband, R., Herre, E.A., Turner, S.L. Gallery, R. and Young, J.P. (2002) Molecular diversity of arbuscular mycorrhizal fungi and patterns of host association over time and space in a tropical forest, *Mol Ecol*, 11: 2669-2678.

Joner, E.L., Briones, R. and Leyval, C. (2000) Metal-binding capacity of arbuscular mycorrhizal mycelium, *Plant Soil* 226: 227-234.

Katoh, K., Misawa, K., Kuma, K. and Miyata, T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform, *Nucleic Acids Res*, 30(14): 3059-3066.

Katoh, K. and Standley, D.M. (2013) MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability, *Mol Biol Evol*, 30(4): 772-780.

Kimura, M. (1980) A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences, *Journal of Molecular Evolution*, 16:111-120.

Knight, R., Maxwell, P., Birmingham, A., Carnes, J., Caporaso, J. G., Easton, B. C., … Huttley, G. A. (2007) PyCogent: a toolkit for making sense from sequence. *Genome Biology*, 8(8): R171, http://doi.org/10.1186/gb-2007-8-8-r171

Krüger, M., Krüger, C., Walker, C., Stockinger, H. and Schüßler, A. (2012) Phylogenetic reference data for systematics and phylotaxonomy of arbuscular mycorrhizal fungi from phylum to species level, *New Phytologist*, 193: 970-984.

Kuhn, G., Hijri, M. and Sanders, I.R. (2001) Evidence for the evolution of multiple genomes in arbuscular mycorrhizal fungi, *Nature*, 414: 745-748.

Kumar, S., Stecher, G. and Tamura, K. (2015) MEGA7: Molecular Evolutionary Genetics Analysis version 7.0 for bigger datasets, *Molecular Biology and Evolution*, (submitted at time of thesis printing).

Lan, Y., Morrison, J.C., Hershberg, R., and Rosen, G.L. (2014) POGO-DB—a database of pairwise-comparisons of genomes and conserved orthologous genes, *Nucleic acids research*, 42(D1): D625-D632.

Larsen, M.V., Cosentino, S., Rasmussen, S., Friis, C., Hasman, H., Marvig, R.L., Jelsbak, L., Sicheritz-Pontén, T., Ussery, D.W., Aarestrup, F.M. and Lund, O. (2012) Multilocus Sequence Typing of Total-Genome-Sequenced Bacteria, J. Clin, Microbiol, 50(4): 1355-1361.

Law, R. (1985) Evolution in a mutualistic environment. In: Boucher, D.H., editor. *The biology of mutualism: ecology and evolution*. London: Croons Helm, 145-170.

Lee, E., Eo, J., Ka, K. and Eom, A. (2013) Diversity of Arbuscular Mycorrhizal Fungi and Their Roles in Ecosystems, *Microbiology*, 41(3): 121-125.

Lindahl, B.D., Nilsson, R.H., Tedersoo, L., Abarenkov, K. and Carlsen, T. (2013) Fungal community analysis by high-throughput sequencing of amplified markers – a user's guide, *New Phytologist*, 199: 288-299.

Linderman, R.G. (2000) Effects of mycorrhizas on plant tolerance to diseases. In: Koltai, H. and Kapulnik, Y, editors. *Arbuscular mycorrhizas: physiology and funtion*. Dordrecht: Springer, 345-365.

Lloyd-Macgilp, S.A., Chambers, S.M., Dodd, J.C., Fitter, A.H., Walker, C. and Young, J.P.W. (1996) Diversity of the ribosomal internal transcribed spacers within and among isolates of *Glomus mosseae* and related mycorrhizal fungi, *New Phytologist*, 133: 103-111.

Lozupone, C. and Knight, R. (2005) UniFrac: a New Phylogenetic Method for Comparing Microbial Communities, *Appl Environ Microbiol*, 71(12): 8228-8235.

Ludwig, W., Strunk, O. and Westram, R. (2004) ARB: a software environment for sequence data, *Nucleic Acid Res*, 32: 1363-1371.

Maldonado-Mendoza, I.E., Dewbre, G.R. and Harrison, M.J. (2001) A phosphate transporter gene from the extra-radical mycelium of an arbuscular mycorrhizal fungus *Glomus intraradices* is regulated in response to phosphate in the environment, *Mol Plant Microbe Interact*, 14: 1140-1148.

Martin, M. (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads, *EMBnet.journal*, 17(1): http://dx.doi.org/10.14806/ej.17.1.200.

Masella, A.P., Bartram, A.K., Truszkowski, J.M., Brown, D.G. and Neufeld, J.D. (2012) PANDAseq: paired-end assembler for illumina sequences, *BMC Bioinformatics*, 13:31, doi:10.1186/1471-2105-13-31.

Masoudi-Nejad, A., Tonomura, K., Kawashima, S., Moriya, Y., Suzuki, M., Itoh, M., ... and Goto, S. (2006) EGassembler: online bioinformatics service for large-scale processing, clustering and assembling ESTs and genomic DNA fragments, *Nucleic acids research*, 34(suppl 2): W459-W462.

McKay, D.L. and Blumberg, J.B. (2007) A Review of the Bioactivity of South African Herbal teas: Rooibos (*Aspalathus linearis*) and Honeybush (*Cyclopia intermedia*), *Phytotheraphy Research*, 21: 1-16.

Meyer, F., Paarmann, D., D'Souza, M., Olson, R., Glass, E.M., Kubal, M., ... & Wilkening, J. (2008) The

metagenomics RAST server–a public resource for the automatic phylogenetic and functional analysis of metagenomes, *BMC bioinformatics*, 9(1): 386.

Morton, J.B. (1988) Taxonomy of VA mycorrhizal fungi: classification, nonmenclature, and identification, *Mycotaxon*, 32: 267-324.

Morton, J.B. and Benny, G.L. (1990) Revised classification of arbuscular mycorrizal fungi (Zygomycetes): a new order, Glomales, to new suborders, Glomineae and Gigasporineae, and two newfamilies, Acaulosporaceae and Gigasporaceae, with an emendation of Glomaceae, *Mycotaxon*, 37: 471-491.

Muthukumar, T., Radhika, K.P., Vaingankar, J., D'Souza, J., Dessai, S. and Rodrigues, B.F. (2009) Taxonomy of AM fungi – An Update. In: *Arbuscular Mycorrhizae of Goa – A Manual of Identification Protocols*. Rodrigues, B.F. and Muthukumar, T., Eds., Goa University, Goa.

Nawrocki, E.P., Kolbe, D.L. and Eddy, S.R. (2014) Infernal 1.0: inference of RNA alignments, *Nucleic Acid Res*, 25(10): 1335-1337.

Opik, M., Metsis, M., Daniell, T.J., Zobel, M. and Moora, M. (2009) Large-sclae parallel 454 sequencing reveals host ecological group specificity of arbuscular mycorrhizal fungi in a boreonemoral forest, *New Phytologist*, 184: 424-437.

Opik, M., Davidson, J., Moora, M. and Zobel, M. (2014) DNA-based detection and identificaton of Glomeromycota: the virtual taxonomy of environmental sequences, *Botany*, 92(2): 135-147.

Orsini, M., Carcangiu, S., Cuccuru, G., Uva, P. and Tramontano, A. (2013) The PARIGA Server for Real Time Filtering and Analysis of Reciprocal BLAST Results, *PLOS ONE*, DOI: 10.1371/journal.pone.0062224.

Price, M.N., Dehal, P.S. and Arkin, A.P. (2010) FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments, *PLoS ONE*, 5(3): 1-10.

Pruesse, E., Quast, C. and Knittel, K. (2007) SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB, *Nucleic Acid Res*, 35: 7188-7196.

Rambaldi, D., & Ciccarelli, F.D. (2009) FancyGene: dynamic visualization of gene structures and protein domain architectures on genomic loci, *Bioinformatics*, 25(17): 2281-2282.

Ravnskov, S. and Jakobsen, I. (1995) Functional compatibility in arbuscular mycorrhizas measured as hyphal P transport to the plant, *New Phytologist*, 129: 611-618.

Redecker, D. (2002) Molecular identification and phylogeny of arbuscular mycorrhizal fungi, *Plant Soil*, 244: 67-73.

Redecker, D. and Raab, P. (2006) Phylogeny of the *Glomeromycota* (arbuscular mycorrhizal fungi): recent developments and new gene markers, *Mycologia,* 98: 885-895.

Rosendahl, S. (2008) Communities, populations and individuals of arbuscular mycorrhizal fungi, *New Phytologist,* 178: 253-266.

Rosendahl, S., McGee, P. and Morton, J.B. (2009) Lack of global population genetic differentiation in the arbuscular mycorrhizal fungus *Glomus mosseae* suggests a recent range expansion which may have coincided with the spread of agriculture, *Mol Ecol*, 18: 4316-4329.

Sanders, I.R., Alt, M., Groppe, K., Boller, T. and Wiemken, A. (1995) Identification of ribosomal DNA polymorphisms among and within spores of the Glomales: application to studies on the genetic diversity of arbuscu- lar mycorrhizal communities, *New Phytologist*, 130: 419-427.

Sanger, F., Nicklen, S. and Coulson, A.R. (1977) DNA sequencing with chain-terminating inhibitors, *Proceedings of the National Academy of Sciences*, 74: 5463-5467.

Santamaria, M., Fosso, B., Consiglio, A., De Caro, G., Grillo, G., Licciulli, F., Liuni, S., Marzano, M., Alonso-Alemany, D., Valiente, G. and Pesole, G. (2012) Reference databases for taxonomic assignment in metagenomics, *Briefings in Bioinformatics*, doi:101093/bib/bbs036.

Schiex, T., Gouzy, J., Moisan, A. and Oliveira, Y. (2003) FrameD: A flexible program for quality check and gene prediction in prokaryotic genomes and noisy matured eukaryotic sequences, *Nucleic Acids Res*, 31(13):3738-3741.

Schloss, P.D. (2008) Evaluating different approaches that test whether microbial communities have the same structure, *The ISME Journal*, 2: 265-275.

Schloss, P.D., Westcott, S.L., Ryabin, T., Hall, J.R., Hartmann, M., Hollister, E.B., Lesniewski, R.A., Oakley, B.B., Parks, D.H., Robinson, C.J., Sahl, J.W., Stres, B., Thallinger, G.G., Van Horn, D.J. and Weber, C.F. (2009) Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities, *Appl Environ Microbiol*, 75(23): 7537-7541.

Schüßler, A., Gehrig, H., Schwarzott, D. and Walker, C. (2001) A new fungal phylum, the *Glomeromycota*: phylogeny and evolution, *Mycological Research,* 105: 1413-1421.

Schüßler, A. and Walker, C. (2010) The Glomeromycota: a species list with new families and new genera, Edinburg and Kew: The Royal Botanic Garden Kew, Botanische Staatssammlung Munich, and Oregaon State University.

Schüßler, A., Schwarzott, D. and Walker, C. (2001) A new fungal phylum, the Glomeromycota: phylogeny and evolution, *Mycol Res*, 105: 413-421.

Selosse, M.A. and Rousset, F. (2011) The Plant-Fungal Marketplace**,** *Science,* 333**:** 828-829**.**

Sheneman, L., Evans, J. and Foster, J.A. (2006) Clearcut: a fast implementation of relaxed neighbor joining, *Bioinformatics*, 22(22): 2823-2824.

Simon, L., Lalonde, M. and Bruns, T.D. (1992) Specific amplification of 18S fungal ribosomal genes from vesicular-arbuscular endomycorrhizal fungi colonizing roots, *Appl. Environ. Microbiol*, 58: 291-295.

Smith, S.E. and Read, D. (2008) *Mycorrhizal Symbiosis,* 2nd ed. UK: Academic Press.

Smith, S.E. and Read, D. (2008) *Mycorrhizal symbiosis*. 3rd ed. San Diego: Academic Press.

Smith, S.E. and Dickson, S. (1997) VA Mycorrhizas: Basic Research Techniques, Glen Osmond: Cooperative Research Centre for Soil and Land Management.

Stürmer, S.L. (2012) A history of the taxonomy and systemetics of arbuscular mycorrhizal fungi belonging to the phylum *Glomeromycota*, *Mycorrhiza,* 22: 247-258.

Tcherepanov, V., Ehlers, A., & Upton, C. (2006) Genome Annotation Transfer Utility (GATU): rapid annotation of viral genomes using a closely related reference genome, *BMC genomics*, 7(1): 150.

The NIH HMP Working Group, Peterson, J., Garges, S., Giovanni, M., McInnes, P., Wang, L., … Guyer, M. (2009) The NIH Human Microbiome Project, *Genome Research*, 19(12): 2317–2323.

Tonge, D.P., Pashley, C.H. and Gant, T.W. (2014) Amplicon-Based Metagenomic Analysis of Mixed Fungal Samples Using Proton Release Amplicon Sequencing, *PloS ONE*, 9(4): e93849.doi:10.1371/journal.pone.009349.

Ulster Med, J. (2015) 18th Meeting of the Irish Society of Human Genetics, Friday 4th September 2015, *Ulster Med J*, 84(3): 200-214.

Van der Heijden, M.G., Boller, T., Wiemken, A. and Sanders, I.R. (1998) Different arbuscular mycorrhizal fungal species are potential determinants of plant community structure, *Ecology*, 79: 2082-2091.

van der Heijden, M.G., Klironomos, J.N., Ursic, M., Moutoglis, P., Streitwolf-Engel, R., Boller, T. and Sanders, I.R. (1998) Mycorrhizal fungal diversity determines plant biodiversity, ecosystem variability and productivity, *Nature*, 396(6706): 69-72.

Wang, B. and Qui, Y.L. (2006) Phylogenetic distribution and evolution of mycorrhizae in land plants, *Mycorrhiza,* 16: 299-363.

Wang, Q, Garrity, G.M., Tiedje, J.M. and Cole, J.R. (2007) Naïve Bayesian Classifier for Rapid Assignment of rRNA Sequences into the New Bacterial Taxonomy, *Appl Environ Microbiol.* 73(16):5261-5267; doi: 10.1128/AEM.00062-07 [PMID: 17586664].

Wang, Y., Coleman-Derr, D., Chen, G. and Gu, Y.Q. (2015) OrthoVenn: a web server for genome wide comparison and annotation of orthologous clusters across multiple species, *Nucleic Acids Res*, 43(Web Server issue): W78–W84.

Waterhouse, A.M., Procter, J.B., Martin, D.M.A., Clamp, M. and Barton, G.J. (2009) Jalview Version 2 – a multiple sequence alignment editor and analysis workbench, *Bioinformatics*, 25(9): 1189-1191.

Willis, A., Rodrigues, B.F. and Harris, P.J.C. (2012) The Ecology of Arbuscular Mycorrhizal Fungi, *Critical Reviews in Plant Science,* 32: 1-20, DOI: 10.1080/07352689.2012.683375.

Wintersinger, J.A. and Wasmuth, J.D. (2015) Kablammo: an interactive, web-based BLAST results visualizer, *Bioinformatics*, 31(8): 1305-1306.

Wright, S.F. and Upadhyaya, A. (1998) A survey of soils for aggregate stability and glomalin, a glycoprotein produced by hyphae of arbuscular mycorrhizal fungi, *Plant Soil,* 198: 97-107.

Wubet, T., Weiss, M., Kottke, I., Teketay, D. and Oberwinkler, F. (2006) Phylogenetic analysis of nuclear small subunit rDNA sequences suggests that the endangered African Pencil Cedar, *Juniperus procera*, is associated with distinct members of Glomeraceae, *Mycol Res*, 110(Pt 9): 1059-1069.

Xu, Z., & Wang, H. (2007) LTR_FINDER: an efficient tool for the prediction of full-length LTR retrotransposons, *Nucleic Acids Research*, *35*(Web Server issue): W265–W268. http://doi.org/10.1093/nar/gkm286

Yadav, T.C., Pal, R.R., Shastri, S., Jadeja, N.B. and Kapley, A. (2015) Comparative metagenomics demonstrating different degradative capacity of activated biomass treating hydrocarbon contaminated wastewater, *Bioresource Technology: International Conference on Emerging Trends in Biotechnology*, 188: 24-32.

Appendices

Appendix I: QIIME code

In order to process the 18S data, the tutorial from http://qiime.org/1.4.0/tutorials/processing_18S_data.html was used. This tutorial could also be used to analyse mixed 18S/16S data. Most of the steps in this pipeline were identical to that of the standard 16S pipeline, the reference data that include eukaryotic sequences were required for OTU picking, taxonomic assignments and template-based alignment building. The Silva_104_release.tgz archive was downloaded. This dataset was used as reference dataset to be able to analyse the 18S datasets. The 108 release was used in the same manner as the 104 release as described below.

There were four main steps followed during this analysis:

- Picking Operational Taxonomic Units (OTUs),

- Assigning OTUs to taxonomic identity,
- Separating OTU tables according to domain,
- Generating and filtering an alignment to construct a phylogenetic tree.

In order to make OTU tables for the four different sample datasets, the default settings were used with pick_otus.py. Sequences were clustered at 97% identity and new clusters were allowed. A reference based approach was used with the Silva dataset that discarded sequences that did not cluster with the reference set. There was an advantage to this approach in that one could use the reference taxonomic assignments and tree rather than generating new ones, but this resulted in a large portion of the sequences being discarded. In order to pick the OTUs against the Silva reference dataset and also discard any clusters that did not match, the following command was used:

```
pick_otus.py -i 18S_tutorial_sample_seqs.fna -m uclust_ref -C -r
QIIME_files/rep_set/silva_104_rep_set.fasta -o uclust_ref_picked_otus/
```

This generated a text file containing the OTUs as the OTU mapping file in the reference folder. Then an OTU table with taxonomy strings from the reference dataset were built directly by using the following command:

```
make_otu_table.py -i
uclust_ref_picked_otus/18S_tutorial_sample_seqs_otus.txt -t
QIIME_files/taxonomy_mapping/Silva_taxa_mapping_104set_97_otus.txt -o
otu_table_uclust_ref.txt
```

This command generated an OTU table as shown in the example (from the QIIME tutorial) below:

```
# QIIME v1.2.1-dev OTU table
#OTU ID AB019754 AB032231 AB294260 AB294267 AF391990 AY642706
AY854217 DQ521781 DQ809034 DQ889882 EF018454 EF100339 EU284615
Consensus Lineage
1347 0 1 0 0 0 0 0 0 0 0 0 0 0
Eukaryota;Parabasalia;Trichonymphida;Teranymphidae;Eucomonympha;;Euco
monympha
3838 0 0 0 0 0 0 0 0 0 0 0 0 1 Archaea;Crenarchaeota;Soil
5518 0 0 0 1 0 0 0 0 0 0 0 0 0 Archaea;Crenarchaeota;AK31;uncultured
12442 0 0 0 0 1 0 0 0 0 0 0 0 0 Archaea;Crenarchaeota;Miscellaneous
13457 0 0 0 0 0 0 0 0 0 1 0 0 0
Bacteria;Proteobacteria;Alphaproteobacteria;Rhodospirillales;Rhodospi
rillaceae;uncultured;uncultured
25010 0 0 0 0 0 0 1 0 0 0 0 0 0
Eukaryota;Metazoa;Nematoda;Chromadorea;Desmodorida;Desmodoridae;Spiri
niinae;Spirinia;;Spirinia
```

```
25076 0 0 0 0 0 1 0 0 0 0 0 0 0
Eukaryota;Fungi;Chytridiomycota;environmental
38221 0 0 0 0 0 0 0 0 0 0 0 1 0 Eukaryota;environmental
39680 0 0 0 0 0 0 0 0 0 0 0 1 0 0
Bacteria;Proteobacteria;Deltaproteobacteria;Myxococcales;Sorangiineae
;Polyangiaceae;Sorangium;uncultured
43905 0 0 1 0 0 0 0 0 0 0 0 0 0 Archaea;Crenarchaeota;South
78139 0 0 0 0 0 0 0 0 1 0 0 0 0
Bacteria;Bacteroidetes;Bacteroidia;Bacteroidales;Bacteroidaceae;Bacte
roides;uncultured
88700 1 0 0 0 0 0 0 0 0 0 0 0 0
Archaea;Euryarchaeota;Thermoplasmata;Marine
96301 0 0 0 0 0 0 0 1 0 0 0 0 0
Archaea;Euryarchaeota;Thermoplasmata;Thermoplasmatales;Marine
```

In order to generate *de novo* OTUs, the following command was used:

```
pick_otus.py –I 18S_tutorial_sample_seqs.fna –o uclust_picked_otus/
```

This created a text file in the uclust_picked_otus directory and could then be used to make a representative sequence set for the downstream analysis of the sequences. The following command was used to do this:

```
pick_rep_set.py –I uclust_picked_otus/18S_tutorial_sample_seqs_otus.txt –f
18S_tutorial_sample_seqs.fna –o rep_set.fna
```

In order to assign the taxonomies to the *de novo* OTUs that were previously generated, the command that follows was used. The memory usage of this command can be specified by using the –rdp_max_memory option which has a higher memory usage for retraining taxonomy mapping files.

```
assign_taxonomy.py –I rep_set.fna –o rdp_assigned_taxonomy/ -t
QIIME_files/taxonomy_mapping/Silva_RDP_taxa_mapping.txt –r
QIIME_files/rep_set/silva_104_rep_set.fasta –rdp_max_memory 2000
```

If BLAST is the preferred method for assignments, the generic taxonomic mapping files can be used and the following command will be used:

```
assign_taxonomy.py -i rep_set.fna -o blast_assigned_taxonomy/ -t
QIIME_files/taxonomy_mapping/Silva_taxa_mapping_104set_97_otus.txt -r
QIIME_files/rep_set/silva_104_rep_set.fasta -m blast
```

An OTU table was built that included the taxonomic assignments. The following command was used:

```
make_otu_table.py -i uclust_picked_otus/18S_tutorial_sample_seqs_otus.txt -
t rdp_assigned_taxonomy/rep_set_tax_assignments.txt -o otu_table.txt
```

In order to separate the OTU tables (when working with mixed 16S/18S samples) according to the representative domains, the *split_otu_table_by_taxonomy.py* module was used.

When doing alignments and tree building, the Silva 104 reference dataset was used. First an alignment was created with the *align_seqs.py* module where the core Silva aligned set was used as the template. The following command was used with the representative dataset which was created in the OTU picking step above to do this alignment:

```
align_seqs.py -i rep_set.fna -t
QIIME_files/core_aligned_set/core_Silva_aligned.fasta -o pynast_aligned/
```

The alignment was then filtered where QIIME had incorporated an entropy and gap calculation to the *filter_alignment.py* module which removed the need for the use of a Lanemask. The following command was used:

```
filter_alignment.py -i pynast_aligned/rep_set_aligned.fasta -o
pynast_aligned/ -e 0.10 -g 0.80
```

In the command above, the 10% most variable positions that were greater than 80% gaps was removed – as specified by the *–e* and *–g* parameters. When adequate results are obtained, a phylogenetic tree can be built by using the *make_phylogeny.py* module. Trees and OTU tables can then be used in downstream analysis in QIIME in order to determine abundances, diversities, etc.

Table I.1: The QIIME workflow scripts and parameter changes. These scripts can be used with the Silva 104 reference set and the pick_otus_through_otu_table.py module. It is important to modify the qiime_parameters.txt file in order to correctly point to the Silva reference filepath and also to use dynamic filtering rather than the 16S Lanemask used with 16S datasets.

Python module	Parameter changes
pick_otus	otu_picking_method uclust (should be set to uclust_ref if a reference based approach is needed)
pick_otus	refseqs_fp (specify the filepath to the representative Silva 104 set, if reference based approach is needed)
align_seqs	template_fp (specify the core aligned Silva

	104 FASTA file path)
filter_alignment	lane_mask_fp (do not specify a Lanemask filepath)
filter_alignment	allowed_gap_frac 0.999999 (set to 0.80 instead of default)
filter_alignment	entropy_threshold 0.10 (set to 0.10 if not already set)
assign_taxonomy	id_to_taxonomy_fp (specify the taxonomy mapping file path, RDP version if RDP is the method of choice)
assign_taxonomy	reference_seqs_fp (specify the Silva representative set file path)

Appendix II: Mothur code

Mothur will allways receive input files and output files to the directory which it is executed from.

The code described below was from the Mothur tutorial page on how to analyse fungal data. This page can be found at http://www.bio.utk.edu/fesin/FESIN2010/Workshop2010/Amend/Mothur_tutorial.pdf. The following code was changed accordingly.

Initial processing and clean-up of sequences

It is wise to first look at the file that is going to be analysed and make a summary of the data this file contains. The command used for this is the **summary.seqs** command where the FASTA file can be specified. From this summary the shortest, longest and median length reads can be known, as well as the ambiguous base calls that are present in the sequences.

Then the sequences needed to be cleaned. Pat Schloss stated on the website mentioned above that the trim.seqs command is used for cleaning up the sequences. He said that: “The **trim.seqs** command provides the preprocessing features needed to screen and sort pyrosequences. Similar analyses are provided by RDP; here we give you added flexibility and speed. The command will enable you to trim off primer sequences and barcodes, use the barcode information to generate a group file and split a FASTA file into sub-files, screen sequences based on the QUAL file that comes from 454 sequencers, sift sequences based on sequence length and the presence of ambiguous bases, and get the reverse complement of your sequences. While this analysis is clearly geared towards pyrosequencing collections, it can also be used with traditional Sanger sequences.”

An OLIGOS file was created in order to specify the primers that were used during pyrosequencing. The *oligos* option took a file that contained the sequences of the forward and reverse primers and, if needed, the barcodes (tags). The sample identifiers were also added to this file. Each of the lines in the OLIGOS file either started with “forward”, “reverse” or “barcode”. A “#” can be added if the line needs to be ignored. Mothur does not allow less than an exact match in the primer and barcode sequences.

The **trim.seqs** command lead to the creation of new files. The *scrap* file contained sequences that did not pass, the *trim* file contained sequences that did pass, and the *summary* file contained sequence by sequence analysis of what failed and what passed. When the barcode was added to the OLIGOS file, a *groups* file was created that sorted the sequences by barcode into the samples from which they originated.

Again a summary of the trimmed results was done by using the summary.seqs command and specifying the FASTA file to be used as the trim.fasta file.

It is also possible to trim sequences due to quality scores. The 454 platform also creates a QUAL file together with the FASTA file. This QUAL file contained numbers that indicated the quality of each base in the sequence instead of the letters that showed which base it was. When a quality file is available, the *qaverage* or *qthreshold* options can also be used with the trim.seqs command.

Mothur has a built-in aligner that was used in this study. Because the 18S sequences were used, an alignment needed to be created. This was done by using the MaarjAM (Opik *et al*., 2010)

database and MAFFT (Katoh *et al*., 2002; Katoh and Standley, 2013) as multiple sequence alignment algorithm. The command used here was **align.seqs** where the candidate was specified as the FASTA file that was the output from the **unique.seqs** command, and the template was specified as the newly aligned MaarjAM database. "flip=T" was added to the command in order to take care of any reverse complimentary sequences that were found in the alignments.

The command used to do chimera checking was **chimera.slayer**. Both the FASTA file produced as output in the previous step and the template was used as parameters. In order to remove the chimeric sequences from the files, the **remove.seqs** function was used. The parameters for this function was the name, groups and alignment files. The summary.seqs command was used again in order to determine how many sequences were chimeric. Because the alignment contained a lot of gaps, the **filter.seqs** command was used again in order to get rid of all the alignment positions that only contained gap characters.

The programs Jalview (Waterhouse *et al*., 2009) or MEGAN (Huson *et al*., 2007) can be used in order to generate a phylogenetic tree from the alignment files. When working with large datasets, the program ClearCut (Sheneman *et al*., 2006) can be used. For Clearcut there is a wrapper included in Mothur. FastTree (Price *et al*., 2010) can also be used, but it was important to note that the trees generated by this program are not easily compatible with Mothur.

The functions used here are **parsimony** which determines the number of randomizations; and **read.tree** to read the tree, groups and name files into memory. This will produce a significance score which can be used to tell whether the community structure of at least one sample differes from the random. Pairwise comparisons of these samples can also be looked at. This step can be computationally intensive because it increases with exponentially with the number of samples. The parsimony command will give an output summary file in which all the pairwise comparisons can be seen. The statistical aspect of this step is to measure the significance and should not be interpreted as the degree of dissimilarity.

UniFrac is another metric of community phylogenetic dissimilarity. There are two versions of this program. One being the weighted version and the other being the unweighted version. The weighted version, the abundance of sequences is taken into account along with the phylogenetic

similarity, or the shared branch length, in order to calculate the similarity between communities. The unweighted version does not take the sequence abundance into account.

Either the function of **unifrac.weighted** or **unifrac.unweighted** can be used. This will result in a p-value that indicates how many of the samples differ from the randomly generated tree. The groups can then be compared to each other and a distance matrix can also be generated. The output will be the pairwise differences between the communities and the ".dist" file will give information in a distance matrix which can be used for ordinations or multivariate statistics.

There is also a built-in function which can be used to generate principal components (PCoA) ordinations using the distance matrices and the **pcoa** function. This will lead to scatter plots (by only using the first two of the three axes) and also "loadings" files that can be used to determine how much variance was described by each of the axes.

After failure of most of the pipelines during the second stage of the reads analyses, it was thought wise to generate phylogentic trees separately, as discussed in chapter 3.

Appendix III: Optional method nr 1

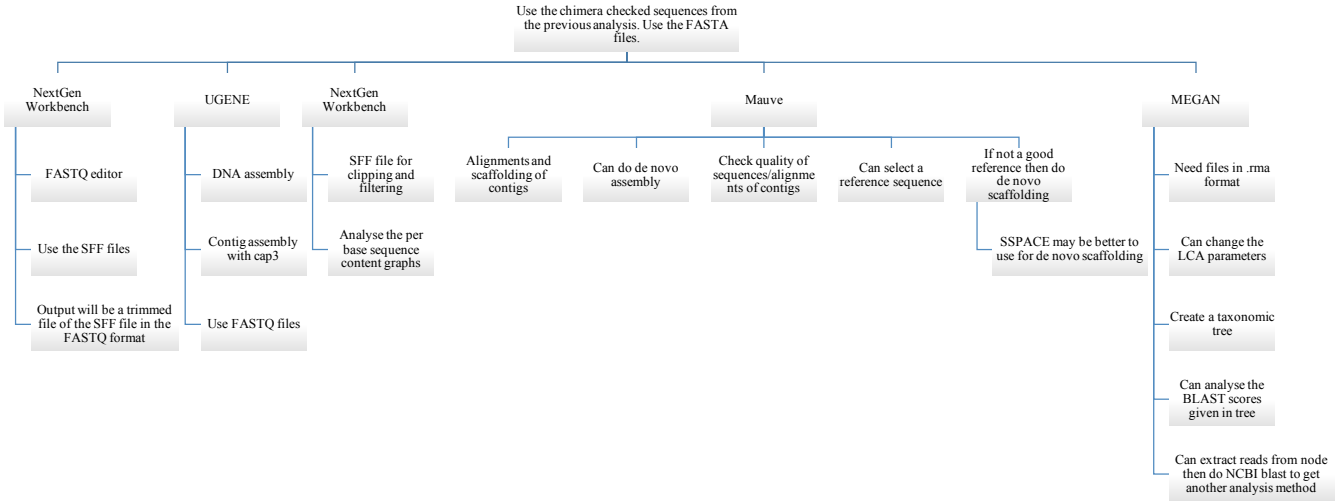


Figure III.1: Overview of the programs that can be used to analyse the results of the different samples containing 18S AM data. This overview is usually used to analyse 16S data, but some changes can be made in future in order to be able to analyse 18S AM data.

The following was tried on NextGen Workbench: The Honeybush natural SFF file was opened and it was seen that there were primers still attached to the sequences. The option for "End clipping & filtering" was used where the settings were:

- Use the adapter clipping info of the SFF file
- Cut 20 bases at the start/end of sequence (5' -/3' end)
- Cut reads shorter than 10 bases
- Cut reads if they contain more than 5% ambiguous ('N') bases
- Cut read if 50% of the bases are either A, C, T or G.

This was repeated with all of the samples FASTA files after the primers were removed in Gedit. The results of the SFF files were disregarded as it showed the exact same thing as the results from the previous QC methods. This QC of the samples after the primers were removed were also done in order to check the quality of the reads after the primers were removed and also to see that there were no primers still present within the reads.

Table III.2: Summary of the results with NextGen Workbench of all the samples. This table showed the total number of reads before and after doing QC with NextGen Workbench, and also the GC percentage of each of the four samples.

Sample	Total reads	Reads after filtering	CG percent
Honeybush natural	17296	17250	1.05%
Honeybush commercial	10175	10150	0.92%
Rooibos natural	16499	16450	1.00%
Rooibos commercial	4438	4400	0.96%

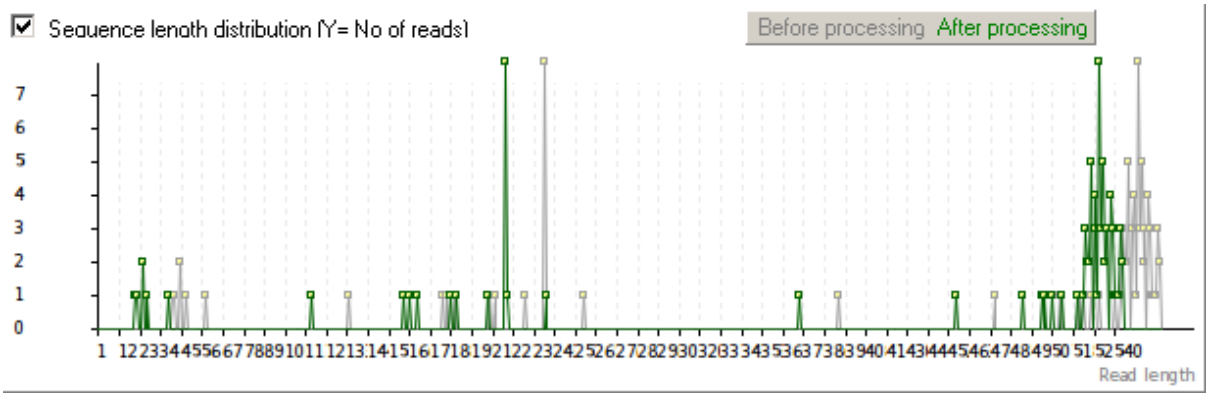


Figure III.2: The sequence length distribution of Honeybush natural sample before and after processing.

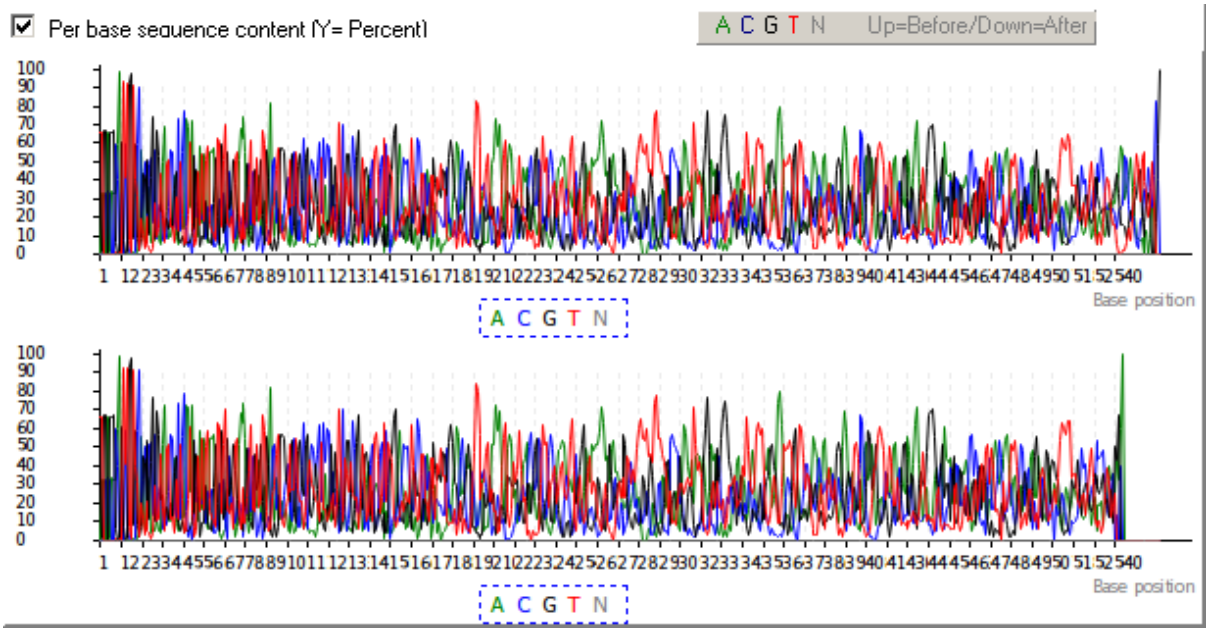


Figure III.3: This showed the Per base sequence content of the Honeybush natural sample. This graph showed the proportion in which ACGT bases were present at each base position. The graph showed the data before and after applying the filters. This can help identify the barcode/recognition sequences if they were not known.

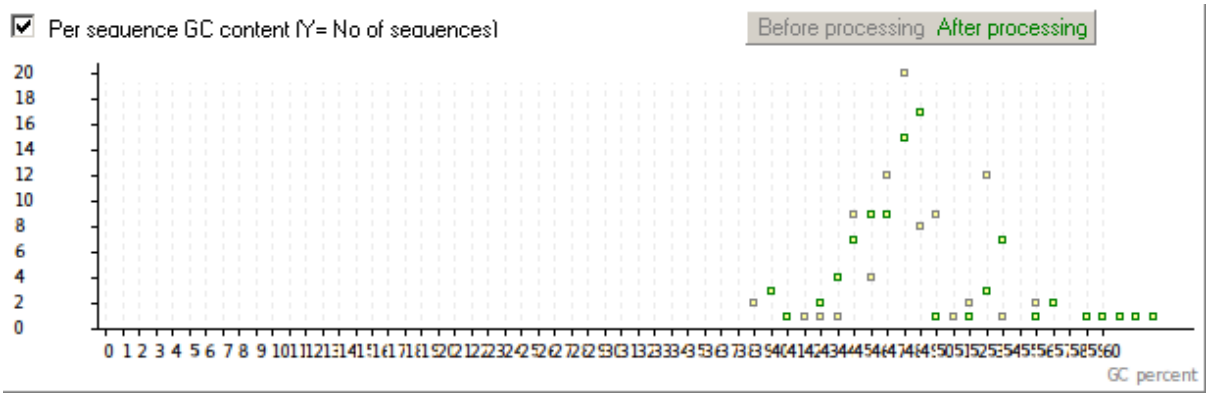


Figure III.4: This figure showed the per sequence distribution of the GC content in the Honeybush natural sample reads before and after processing.

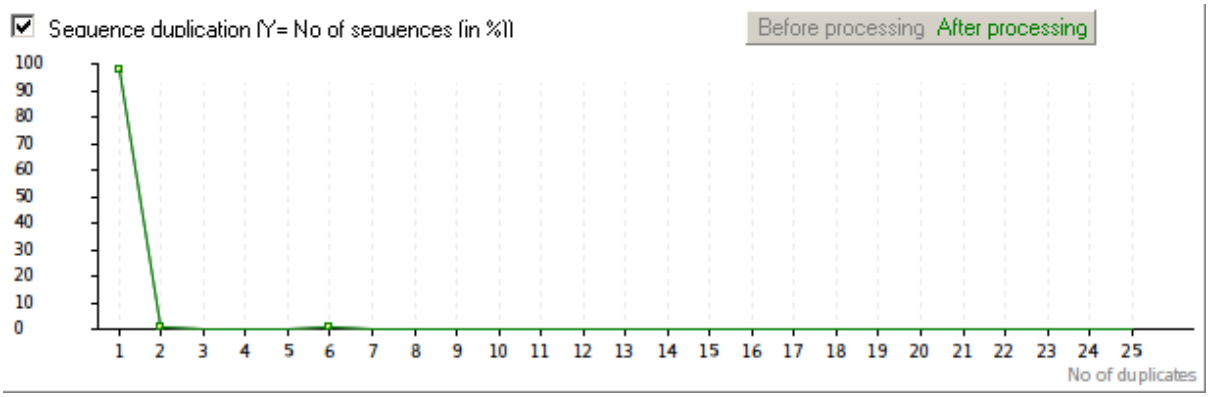


Figure III.5: This figure showed the sequence duplication level in the Honeybush natural reads and how the duplication decreased with processing. The sequence duplication graph showed how many sequences had a duplicate. The x-axis showed the number of duplicates. The y-axis showed how many sequences (percent) have that duplication level. Idealy all sequences should be crowded at the begining of the graph.

There were no overrepresented sequences present in the Honeybush natural sample.

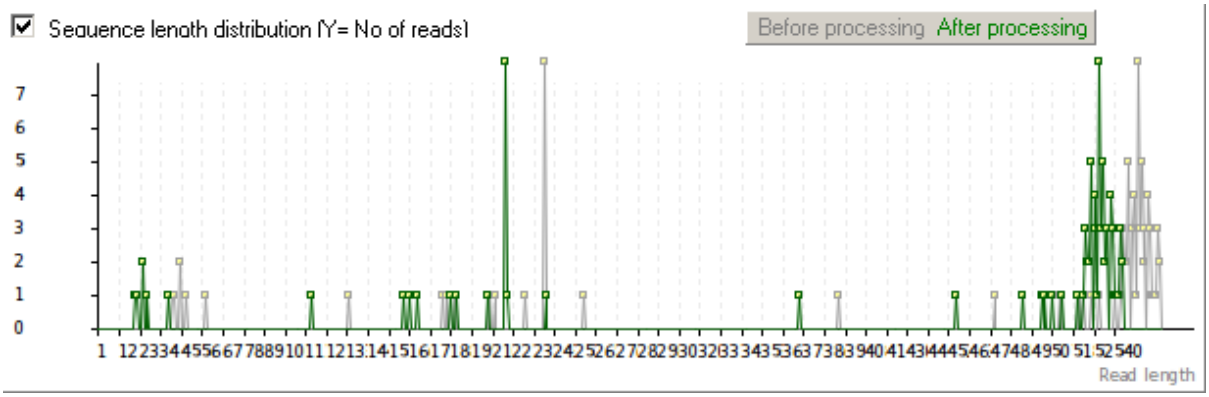


Figure III.6: Representation of the sequence length distribution of Honeybush commercial sample reads.

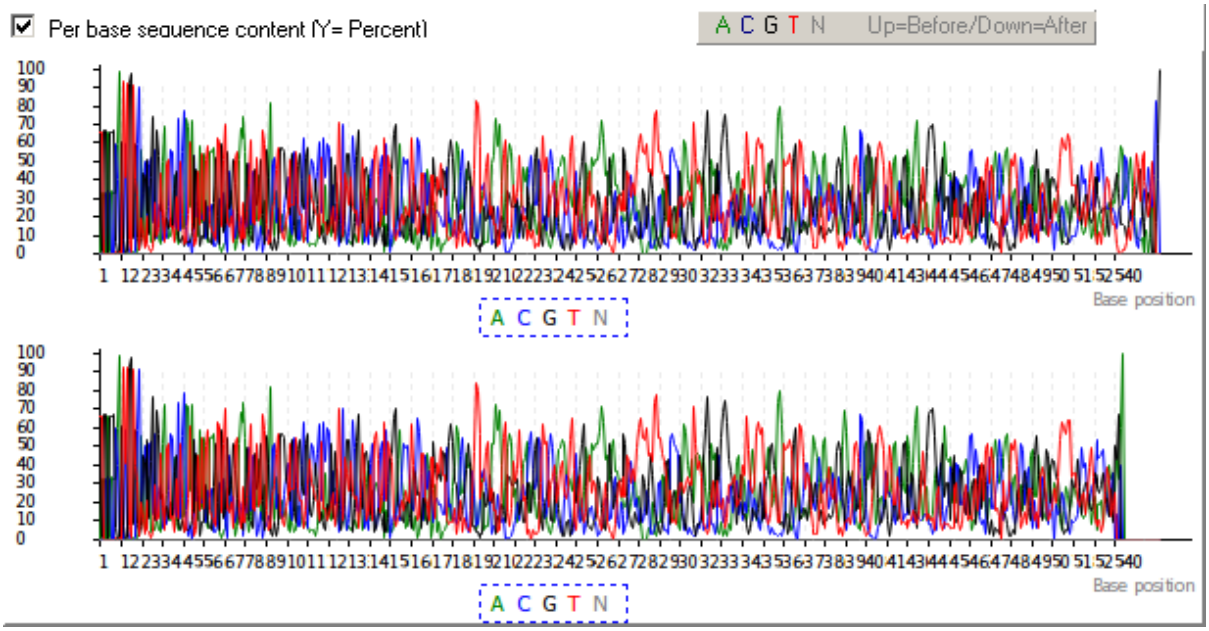


Figure III.7: This showed the per base sequence content of Honeybush commercial sample reads. This graph showed the proportion in which ACGT bases were present at each base position. The graph showed the data before and after applying the filters. This could help identify the barcode/recognition sequences.

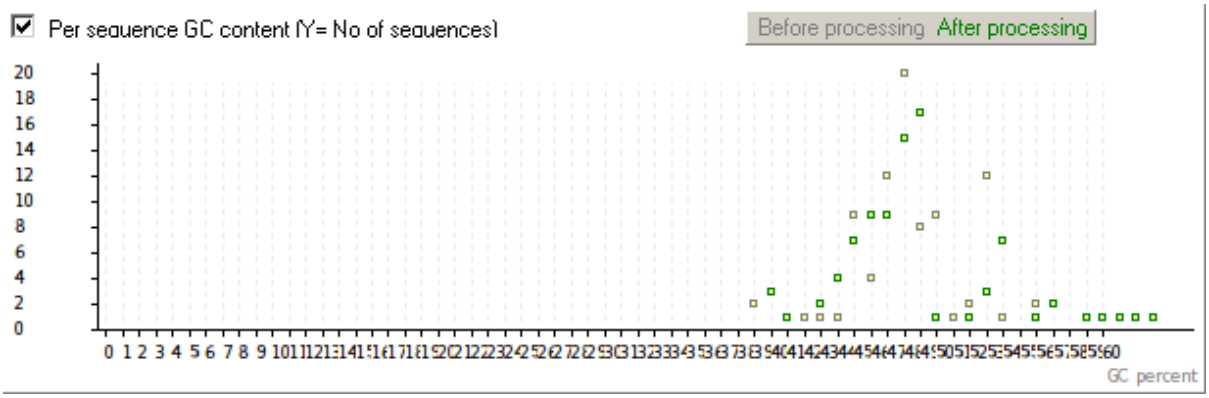


Figure III.8: This showed the per sequence GC content distribution in the Honeybush commercial sample.

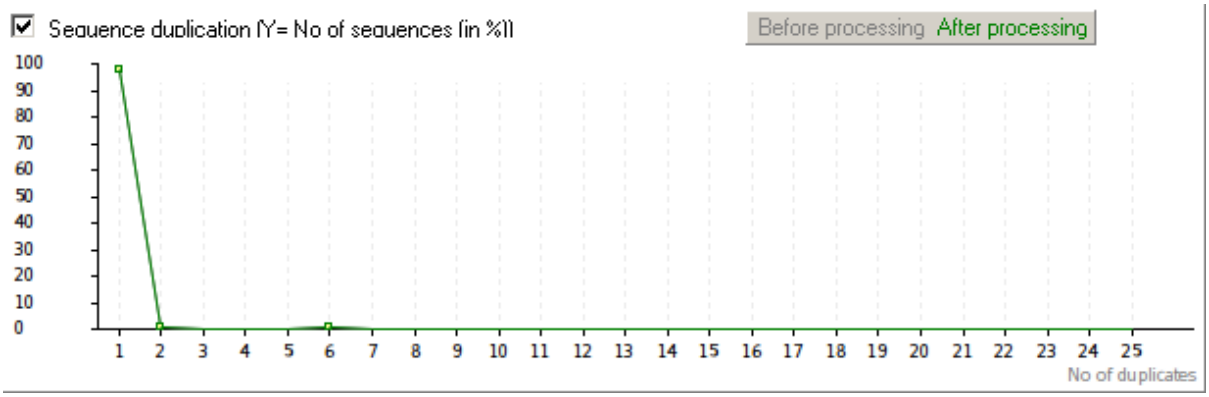


Figure III.9: Representation of the sequence duplication level in the Honeybush commercial sample before and after processing. The sequence duplication graph showed how many sequences had a duplicate. The x-axis showed the number of duplicates. The y-axis

showed how many sequences (percent) had that duplication level. Idealy all sequences should be crowded at the begining of the graph.

Table III.2: This table showed the overrepresented sequences that were present in the Honeybush commercial sample.

Overrepresented sequence	Sequence ID	Count (Before)	Count (After)	Percentage (Before)	Percentage (After)
GAACCCAAACACTTTGGTTTCTCGTAAGGTGCCGAGCGGGTCATCATAGAA ACCCCGCCCGATCCCTAGTCGGCATAGTTTATGGTTAAGACTACGACGGTAT CTGATCGTCTTCGATCCCCTAACTTTCGTTCCCTGATTAATGAAAACATCCT TGGCGAATGCTTTCGCAGTAGTTAGTCTTCAGCAAATCCAAGAATTTCACCT CTGACAGCTGAATACTGACGCCCCCGACTATCCCTATTAATCATTACGGCGG TCCTAGAAACCAACAAAATAGAACCGCACGTCCTATTCTATTATTCCATGCT AATGTATCCGAGCAAAGGCCTGCTTTGAACACTCTAATTTTTTCACAGTAAA AGTCCTGGTTCCCCGCCACAGCCAGTGAAGGCCATGGGATTCCCCAGAAAGA AAGGCCCATCCGGACCAGTACTCGCGGTGAGGCGGACCGGCAGGACGGGCCC AAGGTTCAACTACGAGCTTTTTAACTGCAACAACTTTAATATACGCTATTGG AGCTGGAATTACCGCG	A	0	16	0	0.157
GAACCCAAACACTTTGGTTTCTCGTAAGGTGCCGAGCGGGTCATCATAGAA ACCCCGCCCGATCCCTAGTCGGCATAGTTTATGGTTAAGACTACGACGGTAT CTGATCGTCTTCGATCCCCTAACTTTCGTTCCCTGATTAATGAAAACATCCT TGGCGAATGCTTTCGCAGTAGTTAGTCTTCAGCAAATCCAAGAATTTCACCT CTGACAGCTGAATACTGACGCCCCCGACTATCCCTATTAATCATTACGGCGG TCCTAGAAACCAACAAAATAGAACCGCACGTCCTATTCTATTATTCCATGCT AATGTATCCGAGCAAAGGCCTGCTTTGAACACTCTAATTTTTTCACAGTAAA AGTCCTGGTTCCCCGCCACAGCCAGTGAAGGCCATGGGATTCCCCAGAAAGA AAGGCCCATCCGGACCAGTACTCGCGGTGAGGCGGACCGGCAGGACGGGCCC AAGGTTCAACTACGAGCTTTTTAACTGCAACAACTTTAATATACGCTATTGG AGCTGGAATTACCGCGGCTGCTGGCACCAGACTTCG	B	16	0	0.157	0

When doing a BLAST search for the overrepresented sequences in table III.2 above on the MaarjAM website (http://maarjam.botany.ut.ee/?action=sBlast&method=create), there were various possible hits with maximum identities ranging from 88% to 87%. These are shown in table III.3 and III.4.

Table III.3: Results for BLAST search against the MaarjAM VT database for the overrepresented sequence A specified in table III.2. The results below produced significant alignments.

Accession	Description	Max score	Total score	Query coverage	E-value	Max identity
JX999461	Ambisporaceae Ambispora Shi14a Phy-15 VTX00283	547	547	99%	e-156	88%
AJ301861	Ambisporaceae Ambispora leptoticha VTX00242	537	537	99%	e-153	88%
GU238387	Ambisporaceae Ambispora Amb-1 VTX00283	531	531	99%	e-151	88%
GQ140598	Ambisporaceae Ambispora sp. VTX00242	531	531	99%	e-151	87%
GQ140599	Ambisporaceae Ambispora sp. VTX00242	529	529	99%	e-150	87%
KF467324	Ambisporaceae Ambispora Shi14b Geo-29 VTX00283	523	523	99%	e-148	87%
GU238376	Ambisporaceae Ambispora Amb-1	523	523	99%	e-148	87%

	VTX00283					
AM268193	Ambisporaceae Ambispora fennica VTX00283	523	523	99%	e-148	87%
AM268195	Ambisporaceae Ambispora fennica VTX00283	523	523	99%	e-148	87%
AB047302	Ambisporaceae Ambispora leptoticha VTX00242	521	521	99%	e-148	87%
AB015052	Ambisporaceae Ambispora leptoticha VTX00242	521	521	99%	e-148	87%
KF386269	Paraglomeraceae Paraglomus sp.	515	515	99%	e-146	88%
HE615065	Ambisporaceae Ambispora Torrecillas12b Amb1 VTX00283	515	515	99%	e-146	87%
AB047308	Ambisporaceae Ambispora callosa VTX00242	515	515	99%	e-146	87%
AB047305	Ambisporaceae Ambispora callosa VTX00242	515	515	99%	e-146	87%

Table III.4: Results for BLAST search against the MaarjAM VT database for the overrepresented sequence B specified in table III.2. The results below produced significant alignments.

Accession	Description	Max score	Total score	Query coverage	E-value	Max identity
JX999461	Ambisporaceae Ambispora Shi14a Phy-15 VTX00283	583	583	99%	e-166	88%
AJ301861	Ambisporaceae Ambispora leptoticha VTX00242	573	573	99%	e-163	88%
GU238387	Ambisporaceae Ambispora Amb-1 VTX00283	567	567	99%	e-161	88%
GQ140598	Ambisporaceae Ambispora sp. VTX00242	567	567	99%	e-161	88%
GQ140599	Ambisporaceae Ambispora sp. VTX00242	565	565	99%	e-161	88%
KF467324	Ambisporaceae Ambispora Shi14b Geo-29 VTX00283	559	559	99%	e-159	88%
GU238376	Ambisporaceae Ambispora Amb-1 VTX00283	559	559	99%	e-159	88%
AM268193	Ambisporaceae Ambispora fennica VTX00283	559	559	99%	e-159	88%
AM268195	Ambisporaceae Ambispora fennica VTX00283	559	559	99%	e-159	88%
AB047302	Ambisporaceae Ambispora leptoticha VTX00242	557	557	99%	e-159	88%
AB015052	Ambisporaceae Ambispora leptoticha VTX00242	557	557	99%	e-159	88%
KF386269	Paraglomeraceae Paraglomus sp.	551	551	99%	e-157	88%
HE615065	Ambisporaceae Ambispora Torrecillas12b Amb1 VTX00283	551	551	99%	e-157	88%
AB047308	Ambisporaceae Ambispora callosa VTX00242	551	551	99%	e-157	88%
AB047305	Ambisporaceae Ambispora callosa VTX00242	551	551	99%	e-157	88%

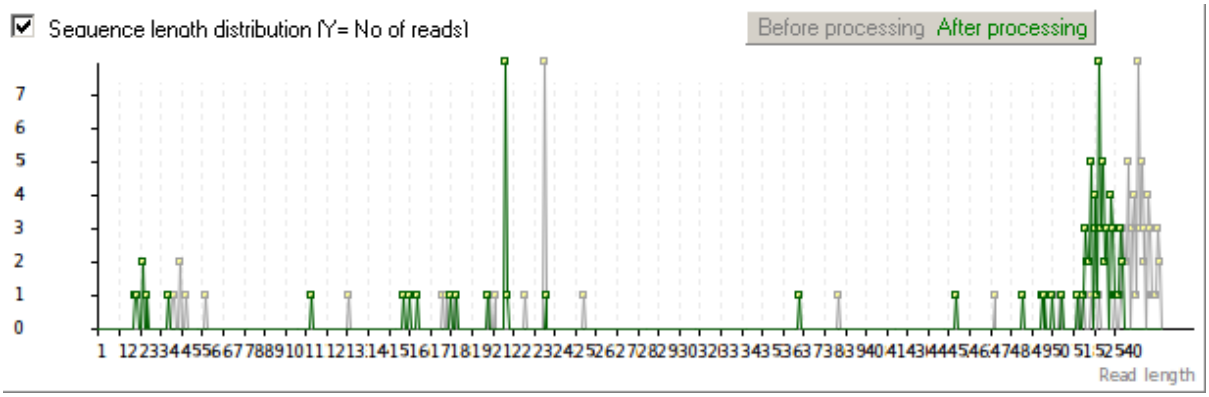


Figure III.10: Sequence length distribution of Rooibos natural sample before and after processing.

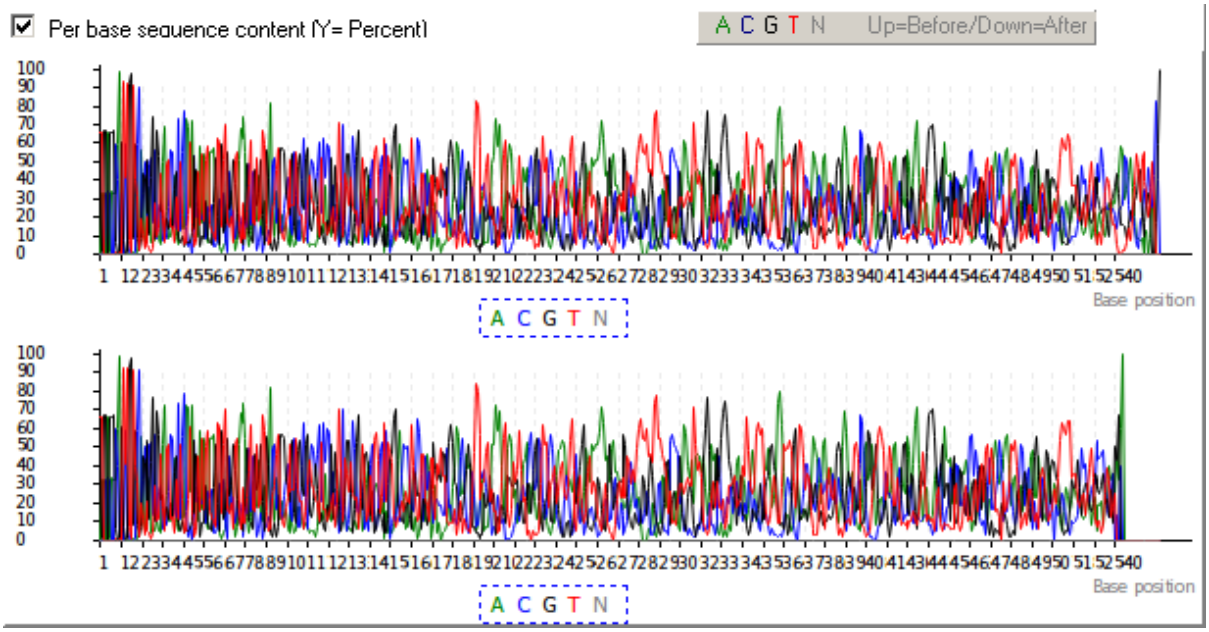


Figure III.11: This figure showed the per base sequence content of the reads in the Rooibos natural sample. This graph showed the proportion in which ACGT bases were present at each base position. The graph showed the data before and after applying the filters. This can help identify the barcode/recognition sequences.

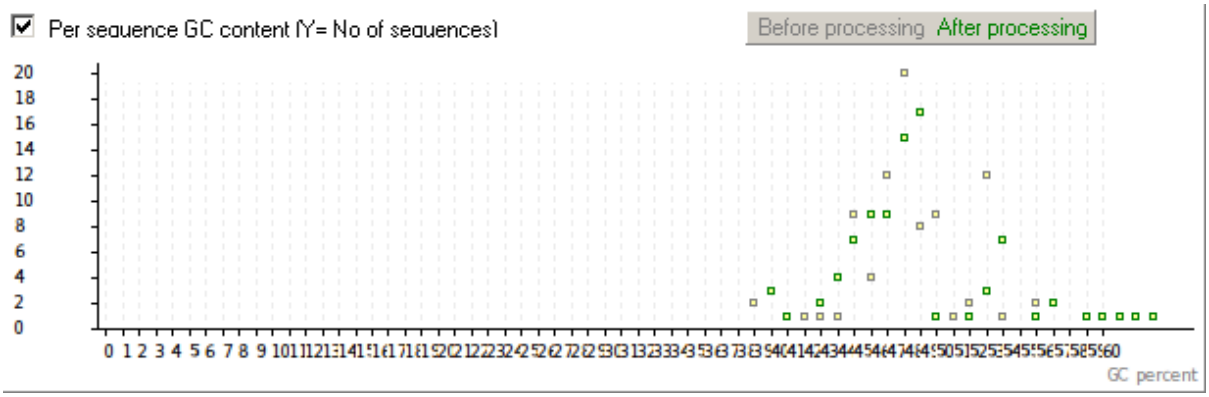


Figure III.12: This figure showed the per sequence GC content of the Rooibos natural sample.

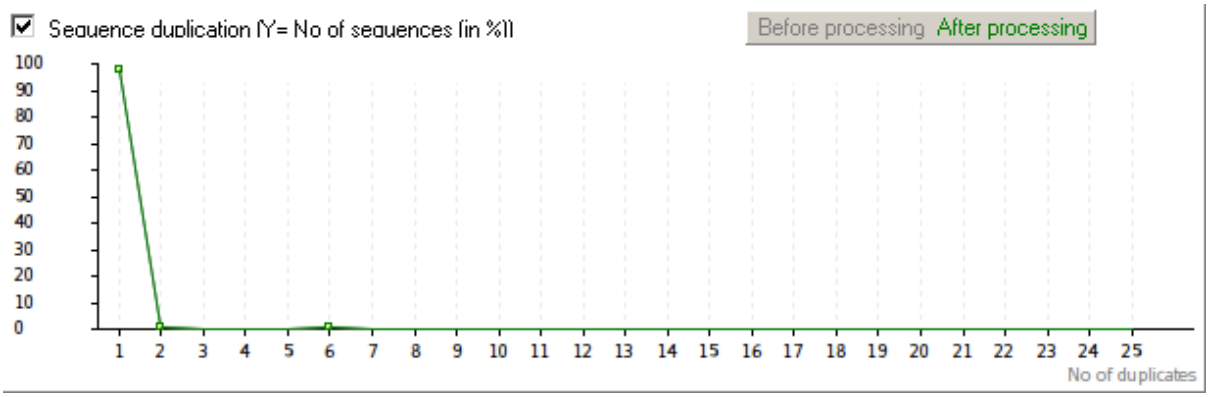


Figure III.13: This figure showed the sequence duplication level of the Rooibos natural sample reads before and after processing. The sequence duplication graph showed how many sequences had a duplicate. The x-axis showed the number of duplicates. The y-axis showed how many sequences (percent) had that duplication level. Idealy all sequences should be crowded at the begining of the graph.

There were no overrepresented sequences present in this sample.

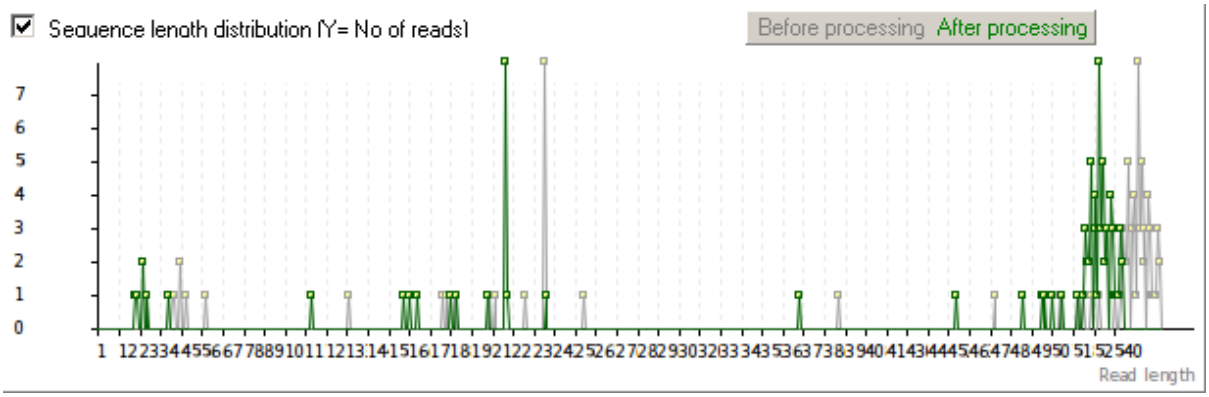


Figure III.14: This figure showed the sequence length distribution of the reads in the Rooibos commercial sample before and after processing.

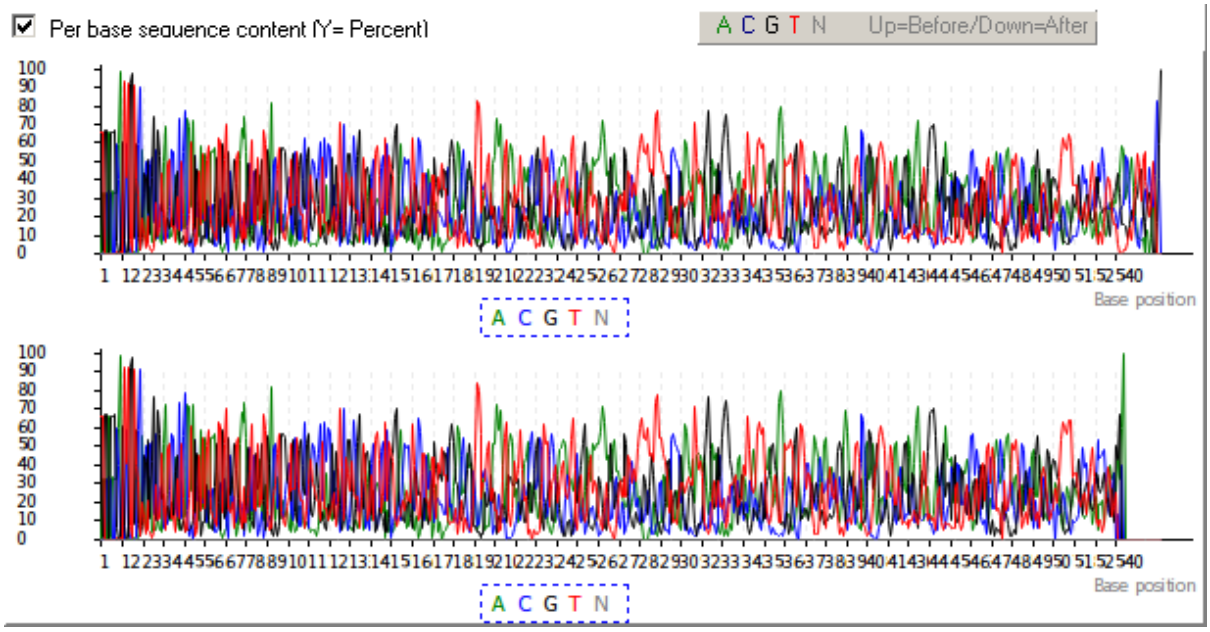


Figure III.15: This figure showed the per base sequence content of the Rooibos commercial sample reads. This graph showed the proportion in which ACGT bases were present at each base position. The graph showed the data before and after applying the filters. This can help identify the barcode/recognition sequences.

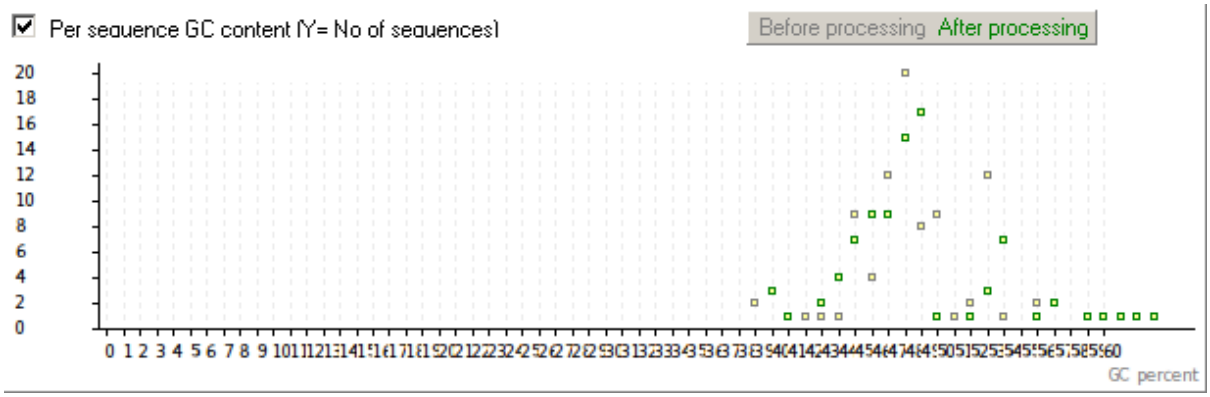


Figure III.16: An illustration of the per sequence GC content of the Rooibos commercial sample reads.

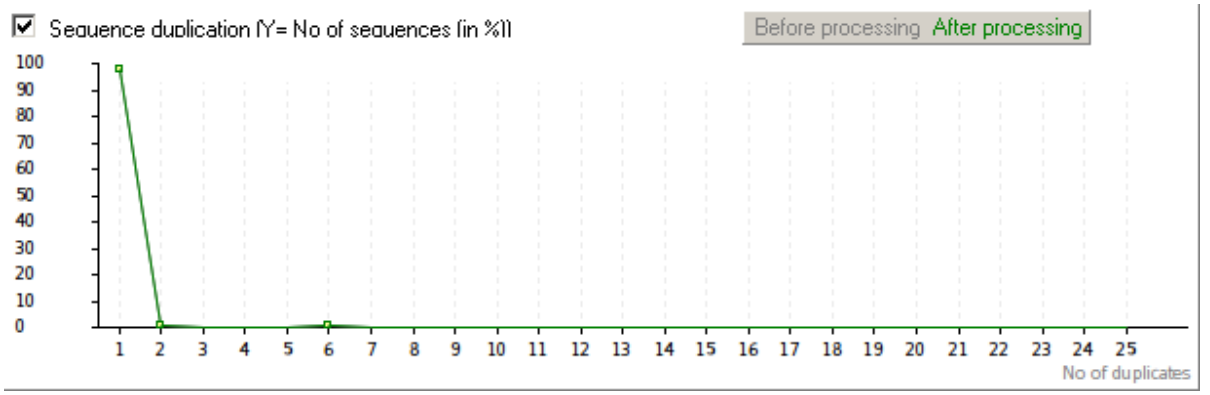


Figure III.17: This figure showed the sequence duplication for the reads in the Rooibos commercial sample before and after processing. The sequence duplication graph showed how many sequences had a duplicate. The x-axis showed the number of duplicates. The y-axis

showed how many sequences (percent) had that duplication level. Idealy all sequences should be crowded at the begining of the graph.

Table III.5: This table gave the overrepresented sequences in the Rooibos commercial sample.

Overrepresented sequence	Sequence ID	Count (Before)	Count (After)	Percentage (Before)	Percentage (After)
TTGGAGGGCAAGTCTGGTGCCATGGGAGGCATGTCAGCTGTTACCCTTACC CGCCAGCATTAGCGCGAAGAGCTCTGCCTGAAAGCGAGCATCATCCAATGC G TTATGGTTCGGTTCGCTGAGTGGCTTCAGGGCAGCGGTCATCTTGGATGATT TTGTTTCCTGCCAGCTGCAACCTACAGCTCCCATGAAGTAGGCCTTCATGTC GATAGCGGTA	A	0	6	0	0.135
TTGGAGGGCAAGTCTGGTGCCATGGGAGGCATGTCAGCTGTTACCCTTACC CGCCAGCATTAGCGCGAAGAGCTCTGCCTGAAAGCGAGCATCATCCAATGC G TTATGGTTCGGTTCGCTGAGTGGCTTCAGGGCAGCGGTCATCTTGGATGATT TTGTTTCCTGCCAGCTGCAACCTACAGCTCCCATGAAGTAGGCCTTCATGTC GATAGCGGTAAAACCAAAGTGTTTGGGTTC	B	6	0	0.135	0

When doing a BLAST search for the overrepresented sequences in table III.5 above on the MaarjAM website (http://maarjam.botany.ut.ee/?action=sBlast&method=create), there were various possible hits with maximum identities ranging from 88% to 87%. These are shown in table III.6 and III.7.

Table III.6: Results for BLAST search against the MaarjAM VT database for the overrepresented sequence A specified in table III.5. The hits below produced significant alignments.

Accession	Description	Max score	Total score	Query coverage	E-value	Max identity
AJ315527	Glomeraceae Glomus cf. microaggregatum	46	46	10%	3E-05	100%
AJ315526	Glomeraceae Glomus cf. microaggregatum	46	46	10%	3E-05	100%
AJ315525	Diversisporaceae Diversispora sp. VTX00054	46	46	10%	3E-05	100%
AJ315524	Diversisporaceae Diversispora sp. VTX00054	46	46	10%	3E-05	100%
HE775382	Glomeraceae Glomus Kohout14 A-6	44	44	10%	0.0001	100%
HE775381	Glomeraceae Glomus Kohout14 A-1	44	44	10%	0.0001	100%
HE775380	Glomeraceae Glomus Kohout14 A-1	44	44	10%	0.0001	100%
KC832518	Glomeraceae Glomus V1-Glom	44	44	10%	0.0001	100%
KC832517	Archaeosporaceae Archaeospora C2-Glom	44	44	10%	0.0001	100%
KC832516	Archaeosporaceae Archaeospora D11-Glom	44	44	10%	0.0001	100%
KC832515	Glomeraceae Glomus C1-Glom	44	44	10%	0.0001	100%
KC832514	Archaeosporaceae Archaeospora D5-Glom	44	44	10%	0.0001	100%
KC832513	Glomeraceae Glomus D10-Glom	44	44	10%	0.0001	100%
KC832512	Glomeraceae Glomus D3-Glom	44	44	10%	0.0001	100%
KC832511	Archaeosporaceae Archaeospora D2-Glom	44	44	10%	0.0001	100%

Table III.7: Results for BLAST search against the MaarjAM VT database for the overrepresented sequence B specified in table III.5. The hits below produced significant alignments.

Accession	Description	Max score	Total score	Query coverage	E-value	Max identity
AJ315527	Glomeraceae Glomus cf. microaggregatum	46	46	9%	3E-05	100%
AJ315526	Glomeraceae Glomus cf. microaggregatum	46	46	9%	3E-05	100%
AJ315525	Diversisporaceae Diversispora sp. VTX00054	46	46	9%	3E-05	100%
AJ315524	Diversisporaceae Diversispora sp. VTX00054	46	46	9%	3E-05	100%
HE775382	Glomeraceae Glomus Kohout14 A-6	44	84	17%	0.0001	100%
HE775381	Glomeraceae Glomus Kohout14 A-1	44	84	17%	0.0001	100%
HE775380	Glomeraceae Glomus Kohout14 A-1	44	84	17%	0.0001	100%
KC832518	Glomeraceae Glomus V1-Glom	44	44	9%	0.0001	100%
KC832517	Archaeosporaceae Archaeospora C2-Glom	44	44	9%	0.0001	100%
KC832516	Archaeosporaceae Archaeospora D11-Glom	44	44	9%	0.0001	100%
KC832515	Glomeraceae Glomus C1-Glom	44	44	9%	0.0001	100%
KC832514	Archaeosporaceae Archaeospora D5-Glom	44	44	9%	0.0001	100%
KC832513	Glomeraceae Glomus D10-Glom	44	44	9%	0.0001	100%
KC832512	Glomeraceae Glomus D3-Glom	44	44	9%	0.0001	100%
KC832511	Archaeosporaceae Archaeospora D2-Glom	44	44	9%	0.0001	100%

Dereplication was only done on the Honeybush commercial and Rooibos commercial samples in order to remove the overrepresented sequences. This was only used for the next step, but when distribution statistics will be done, these overrepresented sequences will be added to the files.

Distance matrices were calculated in UGENE which could then be used in further analysis. The distance matrices were calculated by constructing the matrix according to sequence identity based on the profile that looked at the counts, exclude gaps and showed group statistics of the multiple sequence alignments. The distance matrices were saved as CSV formats to be used in R for statistical analysis.

Appendix IV: Optional method nr 2

Chimeras occur in generated sequences when conserved genes are sequenced. Any contamination of the amplified cluster with several templates of DNA may lead to alterations of the optical signals from the different templates being sequenced (Santamaria *et al.*, 2012). The DECIPHER web-page (http://decipher.cee.wisc.edu/FindChimeras.html/) can be used to remove any chimeras from the sequences. Other chimera removal tools also need to be found and tried seeing that DECIPHER is based on 16S rRNA sequences and not 18S rRNA.

Assembly of NGS reads

Possible base call errors need to be determined. Also base call quality control need to be done. Roche calculates Phred-like quality scores for the different sequences.

NextGen Workbench can be downloaded from www.dnabaser.com/download/SFF%20tools/ and used to edit and convert the SFF formatted sequences produced by 454 pyrosequencing. The SFF files can be opened and quality scores can be calculated and length distribution control can be done, followed by per base sequence quality control to determine at which length the quality will become unstable. Per base sequence content control for artificial polynucleotides can also be done as well as per sequence GC content control for multiple picks that can be followed by sequence duplication control for artificial repeats. All the above mentioned possibilities can be done on both the whole genome sequences as well as the 18S rRNA amplicons. Overrepresented sequences can also be identified (Opik *et al*., 2009). Clipping and filtering of the sequences can be done in NextGen Workbench where the adaptor -, primer - and cut positions can be specified. The above possibilities can then be repeated in order to determine whether the sequences have improved in their quality after clipping and filtering was done in order to exclude low quality reads and duplicates. There is a function to view the individual sequences in the NGS dataset called the Tab Sequence viewer. This can also be used to split (if the files are too big) or convert the sequence files (only in the direction of losing information). If needed multiplexed files can also be split by separating the reads by adapters.

De novo assembly can be studied by using the program UGENE where the DNA assembly tools are used when the contigs are assembled. Re-sequencing assembly can be done by using the program UGENE where the sequences can be aligned to a reference sequence and the short reads can also be aligned. The zooming option can, for instance, be used to adjust the Assembly Browser Window in which an overview of the assembly can be seen between the reference and consensus sequences as well as the coverage of the reads. A comparison between the *de novo*

and re-sequencing results can be done but it is important to note that the *de novo* assembly of the short reads may not be accurate and the re-sequencing process needs a good reference sequence to build the reads.

After the *de novo* and re-sequencing processes was done, it would be wise to do some scaffolding where the contigs are ordered by orientation and location within the whole organisms' genome. In the case of the availability of a good reference genome, the scaffolding can be done by aligning the contigs against the reference sequence. Scaffolding software are available at http://omictools.com/scaffolding-c300-p1.html/. Another scaffolding tool to use will be MAUVE where the contigs can also be aligned against a reference sequence. The program will reorder the contigs and save it in a new FASTA file. The output will show clearly where the contig borders are. In order to finish the scaffolding, the contigs are joined by matched read pairs at the ends of the captured gaps and the ordering and orientation of larger assemblies are done relative to each other. Re-sequencing the closing of the uncuptured gaps will be done in order to have no gaps in the final consensus sequence.

Read binning

Binning refers to the identification of the specific species the NGS reads originated from. It is done at different taxonomic ranks and has two different processes: the first is sequence similarity based methods, and the second is sequence composition based methods. Species identification by 18S rRNA can be done with the use of 18S rRNA databases including NCBI18S.

Sequence similarity based binning can be done by using different BLAST tools to compare the fragments against a reference database like NCBI *nr* (non-redundant). This process requires close homologous sequences to the sample that have already been sampled and sequenced (Opik *et al.*, 2009). But as the database grows, this method will become more accurate while the computational cost will increase. MEGAN (Huson *et al*., 2007) can be used to compare the sequences to a reference database using BLAST. This is dependent on close homologs and it is very difficult to measure the accuracy of MEGAN.

Sequence composition based binning uses the sequence composition in order to measure and quantify the sequences. The sequences are then binned so that similar compositions are binned together. The aim here is to determine the taxonomic diversity of the different environmental samples and the success of the similarity-based binning mostly depends on the existence of similarity searches carried out. DNA reads searched against a database may show different

similarity to the reference sequences in the database. The lowest common ancestor (LCA) approach can then be used to assign the reads to different taxonomic levels. LCA is based on BLAST bit scores. The 18S rRNA can be used to get similar sequences but it must be kept in mind that small, reliable sequences can be found but there can be many copies of target genes in one organism that must be accounted for.

Identification of species diversity

There are various existing tools that can be used:

- *EzTaxon*: to cover species that were not cultured that are found in microbial ecological studies (http://www.ezbiocloud.net/eztaxon/)
- *TaxI*: identifying eukaryotes based on DNA barcoded sequences (http://www.evolutionsbiologie.uni-konstanz.de/Software/)
- *FunGene*: identifying species based on functional marker genes within their sequences (http://fungene.cme.msu.edu/)

Species richness analysis

Richness can be analysed by using rarefaction curves. Rarefaction curves (Opik *et al*., 2009) estimate the coverage obtained from sampling and the form of the curve is analysed visually to be one of the following shapes: broken stick shape, geometric, logarithmic or log-normal, each with their own meaning. The expected species richness can be determined according to the Chao1 equation (Chao, 1984).

The RDP pipeline function for the Shannon (under 'Comparative metagenomics with species diversity analysis') and Chao1 index can be used to alternatively calculate the species richness.

Comparative metagenomics with species diversity analysis

Species evenness refers to how equally species are represented within the samples. Shannon's index of species diversity may be used. There are different forms of species diversity namely the broken stick shape, geometric, logarithmic and log-normal.

Comparative metagenomics can be analysed using the Jaccard similarity and distance coefficients and the Sorensen similarity coefficient. These analysis tools are also part of the RDP Pipeline (https://pyro.cme.msu.edu/index.jsp). The Metastats website can also be used to analyse the results statistically. It is a highly published and used website for statistical analysis of metagenome samples. It can be found at http://metagenomics.atc.tcs.com/.

Metagenome databases

Databases that can be of help during developing the pipeline may be the following:

- *MG-RAST*: it is a community resource (http://metagenomics.anl.gov/metagenomics.cgi?page=Home)
- *MeganDB*: it uses an RMA file that is pre-processed metagenome data file (http://www.megan-db.org/megan-db/). It can be opened in MEGAN.

Creating an artificial metagenome

An interesting extra analysis can be done by creating and analysing an artificial metagenome in order to evaluate the results found. This method can be used because it is difficult to evaluate the methods seeing that the exact binning solutions of the available metagenomes are not known. The data used to construct this artificial metagenome will most probably be the same data used to train the algorithm. MetaSIM (http://ab.inf.uni-tuebingen.de/software/metasim/) can be used to create the artificial metagenome.

Appendix V: Optional method nr 3

Alignment of sequence outputs from different initial processing pipelines

The sequences found as output from the different pipelines initial processing were aligned using MAFFT. The MAFFT results are shown below. MAFFT was used in the terminal to align

these sequences because the EBI website (http://www.ebi.ac.uk/Tools/msa/) with the multiple sequence alignment programs, did not allow the alignment of files larger than 1MB. MAFFT (Multiple Alignment using Fast Fourier Transform) uses Fast Fourier Transforms to align the sequences and is suitable for medium to large alignments (http://www.ebi.ac.uk/Tools/services/web_mafft/toolform.ebi). MAFFT is a multiple sequence alignment program for UNIX operating systems and offers a wide range of alignment methods, for example L-INS-I that is accurate with sequence alignments of <~ 200 sequences, and FFT-NS-2 which is best for fast alignments with <~ 30,000 sequences (Katoh and Standley, 2013).

MUSCLE (http://www.ebi.ac.uk/Tools/msa/) was not used as the alignment program because it is mostly used with proteins and medium alignments.

Also, before alignments were started, the sequence files were checked for sequences less than 10 bp long and these sequences were deleted from the files. This was done to remove the bias when aligning with short sequences and when using the sequences to construct a phylogenetic tree, the programs will only classify the sequences up until the shortest sequence in the file.

All these alignment files were viewed in Jalview (Waterhouse *et al*., 2009) to get a comprehensive idea as to what the alignments look like.

Sequence identities were determined for each sample file

The sequence identities for each sample files were determined with the code that Caroline Ross from RUBi provided. Some changes were made to the code and the code used was as follow:

```
import copy
from Bio import pairwise2
from Bio.SubsMat import MatrixInfo as matlist

matrix = matlist.blosum62
gap_open = -3
gap_extend = -0.5
#454Reads.MID7_without_primers.fna
sample ="454Reads.MID10"
file1=open('/home/herna/Desktop/New_comp/MID10/'+sample+'.fna', 'r')
lines = file1.readlines()
file1.close()

reads = {}
header = ''
```

```
sequence=''

#Create dictionary with reference to header of each capsid in file

for line in lines:

    if line.startswith(">"):
        if header != "" and sequence != "":
            reads[header] = sequence
            sequence = ""   #originally not indented
        header = line.rstrip()
    else:
        sequence += line.strip()


if header != "" and sequence != "":
    reads[header] = sequence
    sequence = ""   #originally not indented

Groups={}
GIndex=1

headers = reads.keys()
Groups[1]={headers[0]:reads[headers[0]]}

Groups_ID={}
Groups_ID[1]=[headers[0]+": Reference Seq"]

count=0
for seq in headers[1:]:
    allocated = 0
    for g in Groups:
        g_ID=[]
        if seq in Groups[g]:
            continue
        group_fit = True
        for s in Groups[g]:
            seq1 = reads[seq]
            seq2 = (Groups[g][s])

            #align each chain if each sequence respectively and then concatenate

            chains1 = seq1.split('/')
            #chains2 = seq2.split('/')
            chains2 = ""

            seq1_alinged=''
            seq2_alinged=''

            for i in range(0):

                alns = pairwise2.align.globalds(chains1[i], chains2[i], matrix, gap_open,
gap_extend)
```

```
                top_aln = alns[0]
                seq1_alinged+=top_aln[0]
                seq2_alinged+=top_aln[1]
            #Calculate Sequence Identity
            id_count=0.0
            c=1
            for j in range(len(seq1_alinged)):
                a = seq1_alinged[j].lower()
                b = seq2_alinged[j].lower()

                if a != "-" and b != "-":
                    if a == b:
                        id_count += 1.0
                    if a != "-" or b != "-":
                        c += 1
            seqs_id = (id_count/c)*100
            count+=1


            if seqs_id <85:
                group_fit=False
                continue
            else:
                g_ID.append(seq+" and "+s+": "+str(seqs_id))

        if group_fit:
            Groups[g][seq]= reads[seq]
            Groups_ID[g]+=g_ID
            allocated=1
            break
    if not allocated:
        GIndex+=1
        Groups[GIndex]={seq:reads[seq]}
        Groups_ID[GIndex]=[seq+" and "+s+": "+str(seqs_id)]

print sample
print "Total number os sequences: "+str(len(reads))
print "Number pairwise comparisons: "+str(count)

num_groups= len(Groups)
print "Group Sizes:"
for i in range(1,num_groups+1):
    print str(i)+": "+str(len(Groups[i]))


for pairs in Groups_ID:
    print "Group: "+str(pairs)
    for p in Groups_ID[pairs]:
        print p
'''#Order pairwise ID and calc average similarity per seq in each group
for g in Groups:
    print "Group: "+str(g)
    info = Groups[g]
```

```
#get all pairwise ID
IDList=[]
for seq in info:
    seq_info = info[seq]
    IDList.append(seq_info[1])
print IDList'''
```

The results from this code was found to be inconclusive as to which sequences were the longest and had the highest identities, and so Jalview was used where the sequences were sorted according to pairwise identities.

Sequence outputs from different initial processing pipelines

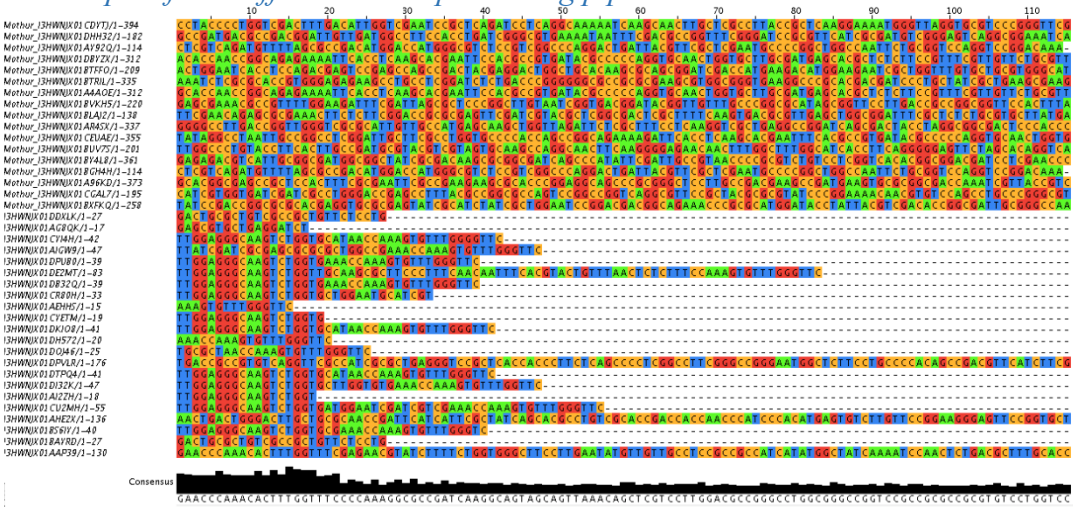


Figure V.1: The Honeybush natural samples sequence outputs from both the Mothur and QIIME pipelines. No alignment with MAFFT was done yet.

In figure V.1 it was seen that the sequences that were found as output after initial processing by Mothur was a lot more like the original sequences used as input for the pipelines. The QIIME pipeline, on the other hand, had sequences as output that were more uniform in the first few bp but this did not continue further down in the sequences. These sequences were also mostly shorter than the original sequences. The same can be seen in the Honeybush commercial, Rooibos natural and Rooibos commercial samples in figures V.2, V.3 and V.4.

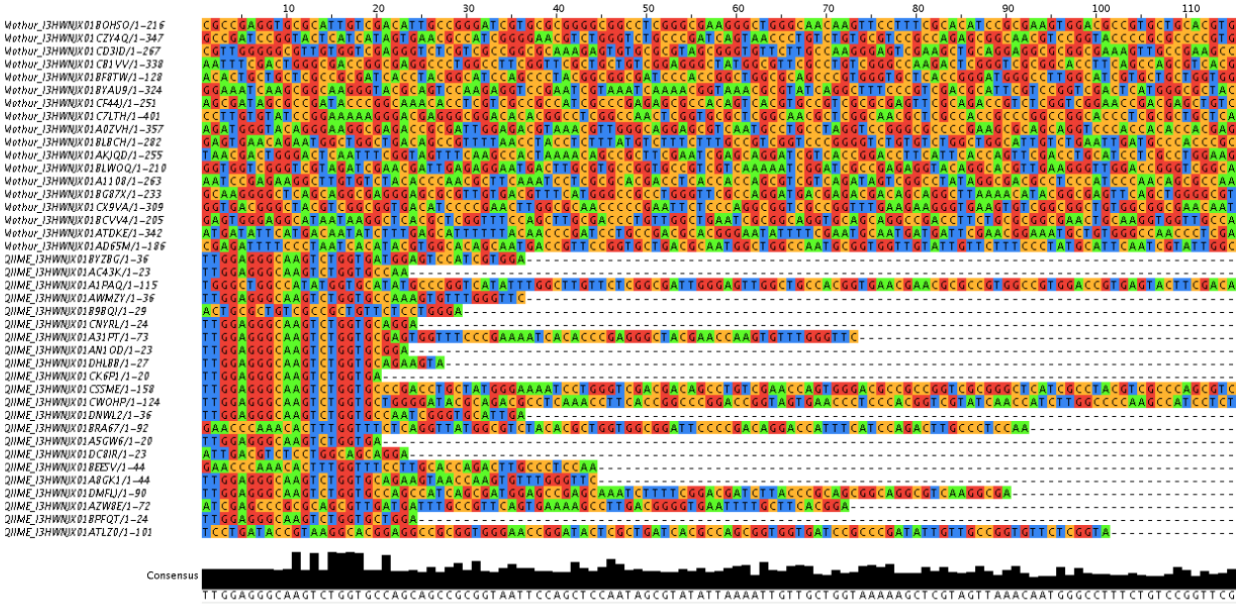


Figure V.2: The Honeybush commercial samples sequence outputs from both the Mothur and QIIME pipelines. No alignment with MAFFT was done yet.

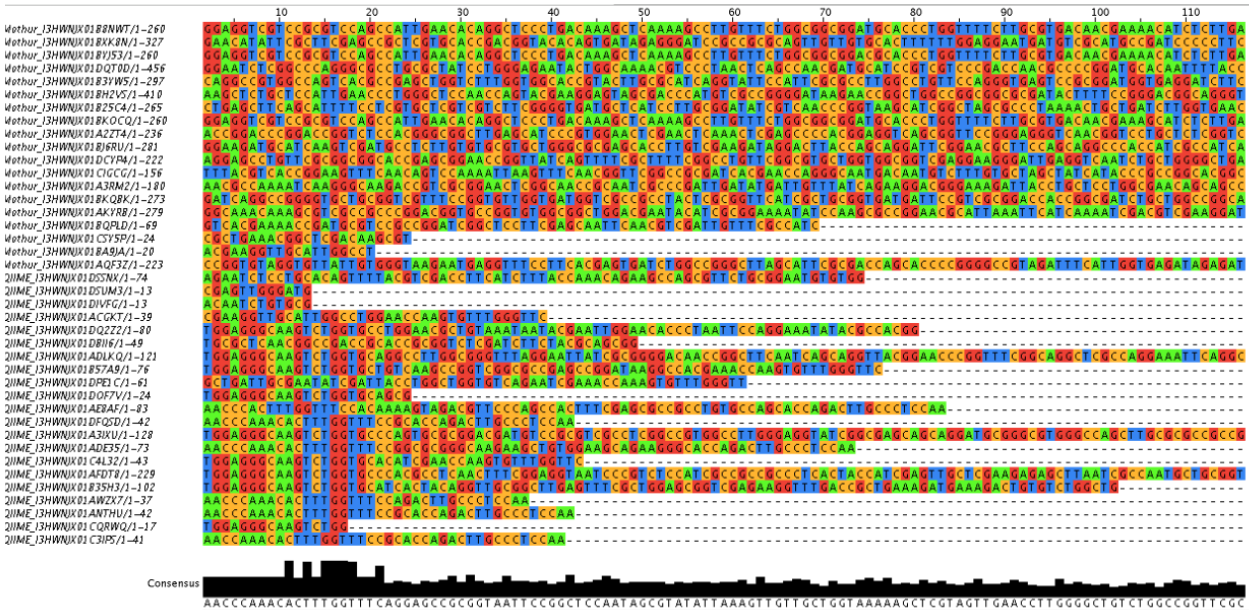


Figure V.3: The Rooibos natural samples sequence outputs from both the Mothur and QIIME pipelines. No alignment with MAFFT was done yet.

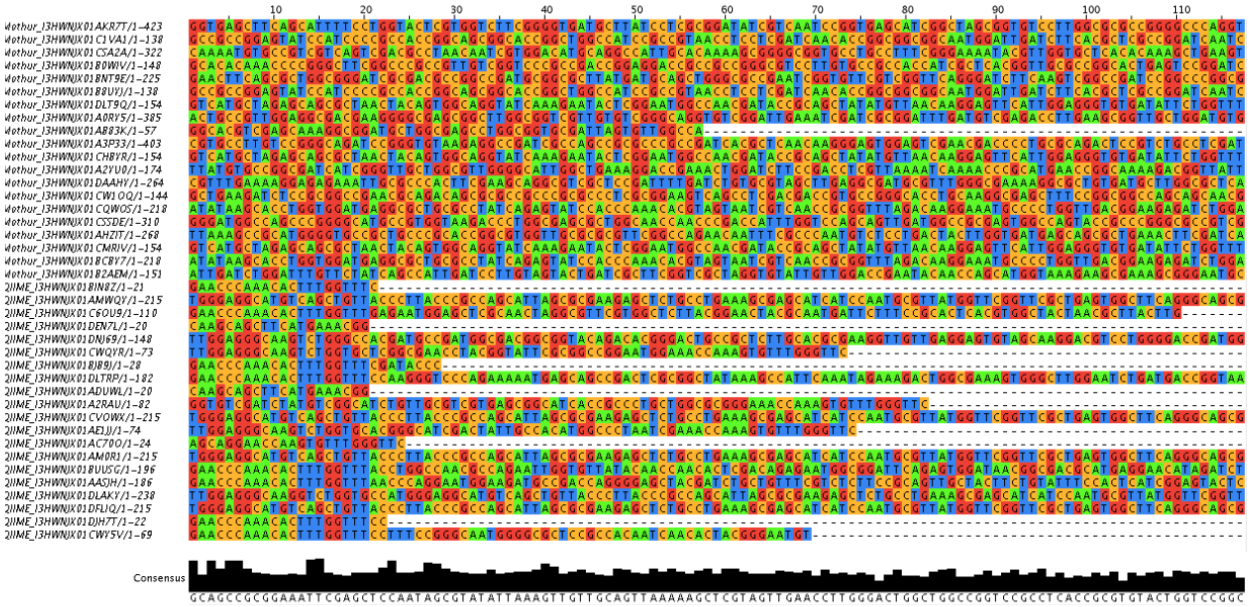


Figure V.4: The Rooibos commercial samples sequence outputs from both the Mothur and QIIME pipelines. No alignment with MAFFT was done yet.

Alignments of sequence outputs from different initial processing pipelines

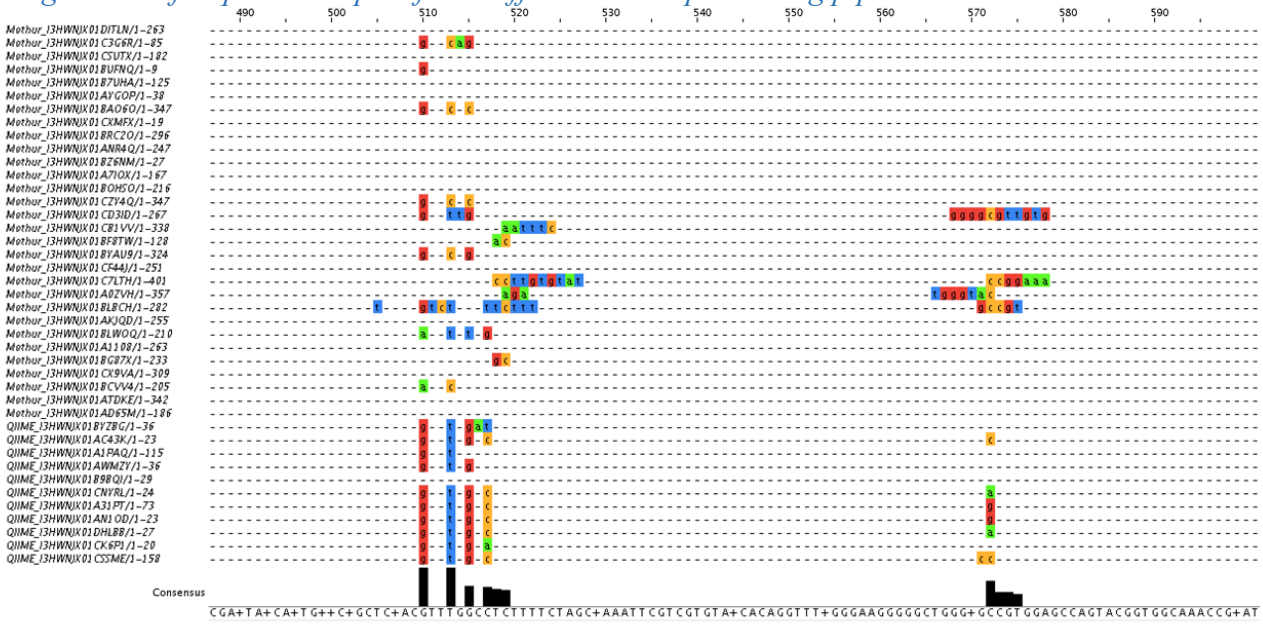


Figure V.5: The Honeybush natural sample sequence output from both the Mothur and QIIME pipelines after initial processing was complete and MAFFT was run to align the sequences. The alignments between the two pipelines for this sample were not good at all.

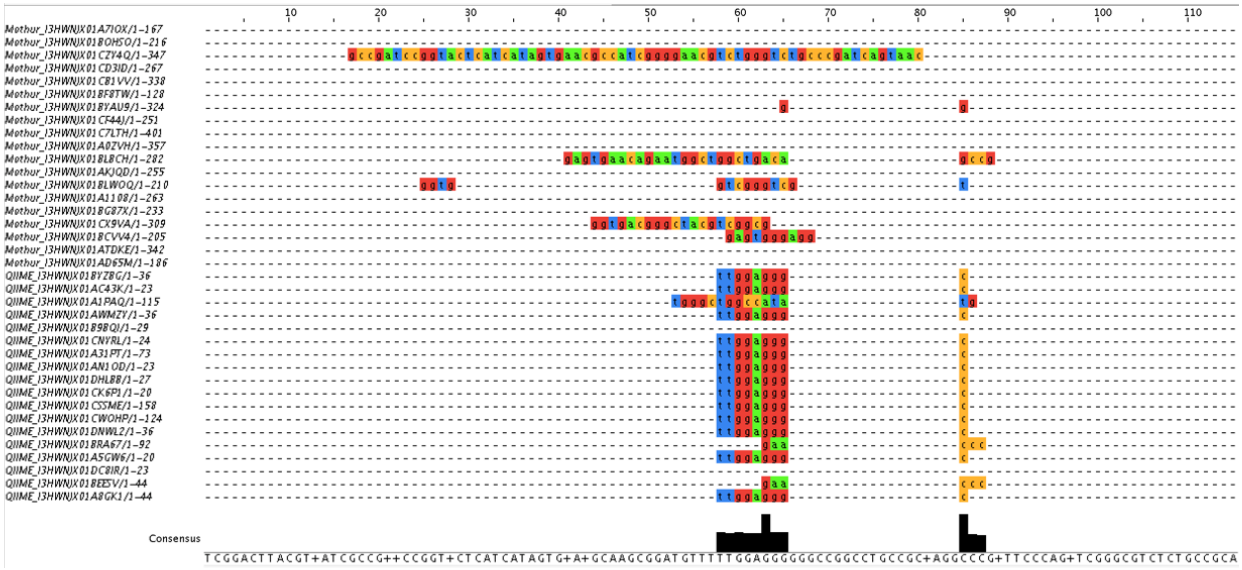


Figure V.6: The Honeybush commercial sample sequence outputs from both Mothur and QIIME after MAFFT was run on the sequences.

The commercial samples were found to have a bit better alignment than the natural samples. For this there is no reason as to why this could have happened – might be because the diversity of the fungi within the commercial samples were a lot less and so the sequences were a bit more easily aligned than the natural sample sequences that possibly contained a higher diversity of fungi. The alignments might refer to regions in the fungal sequences that are conserved throughout the different fungi.

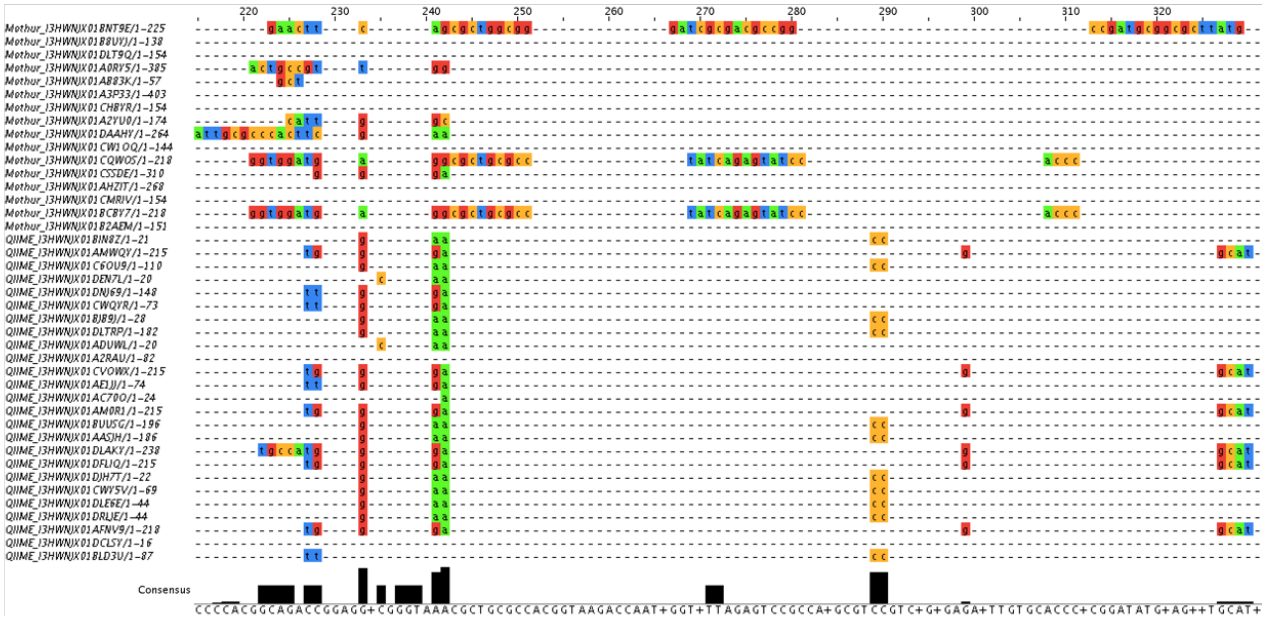


Figure V.7: The Rooibos natural sample sequence output from both Mothur and QIIME after MAFFT was run on the sequences. Some conservation of regions between the different fungi were observed.

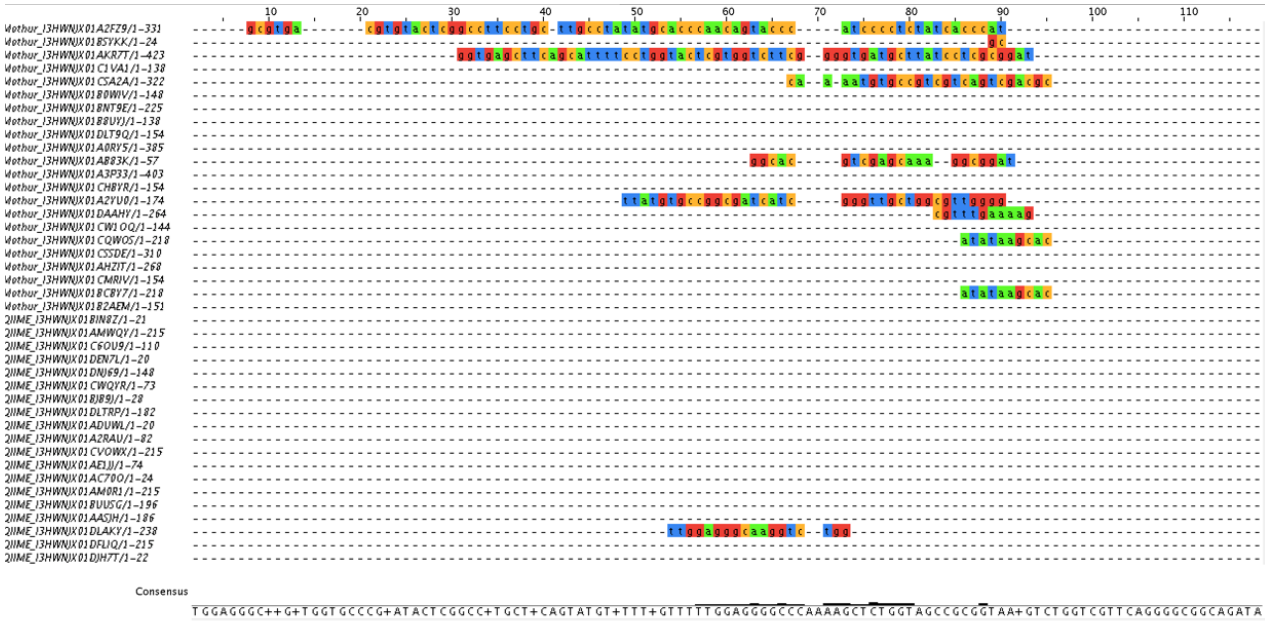


Figure V.8: The Rooibos commercial sample sequence output from both Mothur and QIIME after MAFFT was run on the sequences. This file did not show any conservation of any region. The conclusion made from this figure can be that there were no conserved regions in any of the fungi that were present in this sample.

From the above alignments it was seen that there were no sequences, but rather regions, that were conserved and similar between the different pipelines. This lead to the problem that the pipelines do not give the same results and further studies need to be done in order to determine which of one of these two pipelines will be better to do initial processing of the 18S AM fungal reads with.

Appendix VI: The source code for Cutadapt

The source code used for Cutadapt (Martin, 2011) was found at https://github.com/marcelm/cutadapt/. There is a link to this code on the Python Software Foundation website (https://pypi.python.org/pypi/cutadapt). This code was used as template in order to trim the primer sequences from the sample files during the initial processing step in chapter 2.

Cutadapt is used to find and remove adapters, primers, poly-A tails and unwanted sequences from sequencing data. Reads from some small-RNA reads can contain the 3' adapters because the read is longer than the gene of interest. After high-throughput sequencing, the reads normally start with a primer sequence, and this needs to be trimmed off. Cutadapt can find these adapters and primers in an error-tolerant way. It can also be used to filter the reads in various ways. Cutadapt also comes with a suite of automated tests that can be run on sequencing data (Martin, 2011).

The Python script that follows was found at https://github.com/marcelm/cutadapt/commit/070f415efde2803e0b21b67c672410d36b89d998 and can be downloaded as adapters.py from this website. The raw code was found at https://raw.githubusercontent.com/marcelm/cutadapt/070f415efde2803e0b21b67c672410d36b89d998/cutadapt/adapters.py. This raw code is shown here.

```
# coding: utf-8
"""
Adapters
"""
from __future__ import print_function, division, absolute_import
import sys
import re
from collections import defaultdict
from cutadapt import align, colorspace
from cutadapt.seqio import ColorspaceSequence, FastaReader

# Constants for the find_best_alignment function.
# The function is called with SEQ1 as the adapter, SEQ2 as the read.
# TODO get rid of those constants, use strings instead
BACK = align.START_WITHIN_SEQ2 | align.STOP_WITHIN_SEQ2 | align.STOP_WITHIN_SEQ1
FRONT = align.START_WITHIN_SEQ2 | align.STOP_WITHIN_SEQ2 | align.START_WITHIN_SEQ1
PREFIX = align.STOP_WITHIN_SEQ2
SUFFIX = align.START_WITHIN_SEQ2
ANYWHERE = align.SEMIGLOBAL
LINKED = 'linked'


def parse_braces(sequence):
        """
        Replace all occurrences of ``x{n}`` (where x is any character) with n
        occurrences of x. Raise ValueError if the expression cannot be parsed.

        >>> parse_braces('TGA{5}CT')
        TGAAAAACT
        """
```

```
        # Simple DFA with four states, encoded in prev
        result = ''
        prev = None
        for s in re.split('(\{|\})', sequence):
                if s == '':
                        continue
                if prev is None:
                        if s == '{':
                                raise ValueError('"{" must be used after a character')
                        if s == '}':
                                raise ValueError('"}" cannot be used here')
                        prev = s
                        result += s
                elif prev == '{':
                        prev = int(s)
                        if not 0 <= prev <= 10000:
                                raise ValueError('Value {} invalid'.format(prev))
                elif isinstance(prev, int):
                        if s != '}':
                                raise ValueError('"}" expected')
                        result = result[:-1] + result[-1] * prev
                        prev = None
                else:
                        if s != '{':
                                raise ValueError('Expected "{"')
                        prev = '{'
        # Check if we are in a non-terminating state
        if isinstance(prev, int) or prev == '{':
                raise ValueError("Unterminated expression")
        return result


class AdapterParser(object):
        """
        Factory for Adapter classes that all use the same parameters (error rate,
        indels etc.). The given **kwargs will be passed to the Adapter constructors.
        """
        def __init__(self, colorspace=False, **kwargs):
                self.colorspace = colorspace
                self.constructor_args = kwargs
                self.adapter_class = ColorspaceAdapter if colorspace else Adapter

        def parse(self, spec, name=None, cmdline_type='back'):
                """
                Parse an adapter specification not using ``file:`` notation and return
                an object of an appropriate Adapter class. The notation for anchored
                5' and 3' adapters is supported. If the name parameter is None, then
                an attempt is made to extract the name from the specification
                (If spec is 'name=ADAPTER', name will be 'name'.)

                cmdline_type -- describes which commandline parameter was used (``-a``
                is 'back', ``-b`` is 'anywhere', and ``-g`` is 'front').
                """
                if name is None:
                        name, spec = self._extract_name(spec)
                sequence = spec
                types = dict(back=BACK, front=FRONT, anywhere=ANYWHERE)
                if cmdline_type not in types:
                        raise ValueError('cmdline_type cannot be {0!r}'.format(cmdline_type))
                where = types[cmdline_type]
                if where == FRONT and spec.startswith('^'):  # -g ^ADAPTER
                        sequence, where = spec[1:], PREFIX
                elif where == BACK:
                        sequence1, middle, sequence2 = spec.partition('...')
                        if middle == '...':
                                if not sequence1:  # -a ...ADAPTER
                                        sequence = sequence1[3:]
                                elif not sequence2:  # -a ADAPTER...
                                        sequence, where = spec[:-3], PREFIX
                                else:  # -a ADAPTER1...ADAPTER2
                                        if self.colorspace:
                                                raise NotImplementedError('Using linked adapters
in colorspace is not supported')
                                        if sequence1.startswith('^') or sequence2.endswith('$'):
                                                raise NotImplementedError('Using "$" or "^" when
'
                                                                          'specifying a linked adapter is not
supported')
                                        return LinkedAdapter(sequence1, sequence2, name=name,
```

```
                                        **self.constructor_args)
                elif spec.endswith('$'):   # -a ADAPTER$
                        sequence, where = spec[:-1], SUFFIX
        if not sequence:
                raise ValueError("The adapter sequence is empty.")

        return self.adapter_class(sequence, where, name=name, **self.constructor_args)

    def parse_with_file(self, spec, cmdline_type='back'):
        """
        Parse an adapter specification and yield appropriate Adapter classes.
        This works like the parse() function above, but also supports the
        ``file:`` notation for reading adapters from an external FASTA
        file. Since a file can contain multiple adapters, this
        function is a generator.
        """
        if spec.startswith('file:'):
                # read adapter sequences from a file
                with FastaReader(spec[5:]) as fasta:
                        for record in fasta:
                                name = record.name.split(None, 1)[0]
                                yield self.parse(record.sequence, name, cmdline_type)
        else:
                name, spec = self._extract_name(spec)
                yield self.parse(spec, name, cmdline_type)

    def _extract_name(self, spec):
        """
        Parse an adapter specification given as 'name=adapt' into 'name' and 'adapt'.
        """
        fields = spec.split('=', 1)
        if len(fields) > 1:
                name, spec = fields
                name = name.strip()
        else:
                name = None
        spec = spec.strip()
        return name, spec

    def parse_multi(self, back, anywhere, front):
        """
        Parse all three types of commandline options that can be used to
        specify adapters. back, anywhere and front are lists of strings,
        corresponding to the respective commandline types (-a, -b, -g).

        Return a list of appropriate Adapter classes.
        """
        adapters = []
        for specs, cmdline_type in (back, 'back'), (anywhere, 'anywhere'), (front,
'front'):
                for spec in specs:
                        adapters.extend(self.parse_with_file(spec, cmdline_type))
        return adapters


class Match(object):
    """
    TODO creating instances of this class is relatively slow and responsible for quite
some runtime.
    """
    __slots__ = ['astart', 'astop', 'rstart', 'rstop', 'matches', 'errors', 'front',
'adapter', 'read', 'length']
    def __init__(self, astart, astop, rstart, rstop, matches, errors, front, adapter,
read):
        self.astart = astart
        self.astop = astop
        self.rstart = rstart
        self.rstop = rstop
        self.matches = matches
        self.errors = errors
        self.front = self._guess_is_front() if front is None else front
        self.adapter = adapter
        self.read = read
        # Number of aligned characters in the adapter. If there are
        # indels, this may be different from the number of characters
        # in the read.
        self.length = self.astop - self.astart
        assert self.length > 0
        assert self.errors / self.length <= self.adapter.max_error_rate
```

```
                assert self.length - self.errors > 0

        def __str__(self):
                return 'Match(astart={0}, astop={1}, rstart={2}, rstop={3}, matches={4},
errors={5})'.format(
                        self.astart, self.astop, self.rstart, self.rstop, self.matches,
self.errors)

        def _guess_is_front(self):
                """
                Return whether this is guessed to be a front adapter.

                The match is assumed to be a front adapter when the first base of
                the read is involved in the alignment to the adapter.
                """
                return self.rstart == 0

        def wildcards(self, wildcard_char='N'):
                """
                Return a string that contains, for each wildcard character,
                the character that it matches. For example, if the adapter
                ATNGNA matches ATCGTA, then the string 'CT' is returned.

                If there are indels, this is not reliable as the full alignment
                is not available.
                """
                wildcards = [ self.read.sequence[self.rstart + i:self.rstart + i + 1] for i in
range(self.length)
                        if self.adapter.sequence[self.astart + i] == wildcard_char and
self.rstart + i < len(self.read.sequence) ]
                return ''.join(wildcards)

        def rest(self):
                """
                Return the part of the read before this match if this is a
                'front' (5') adapter,
                return the part after the match if this is not a 'front' adapter (3').
                This can be an empty string.
                """
                if self.front:
                        return self.read.sequence[:self.rstart]
                else:
                        return self.read.sequence[self.rstop:]


def _generate_adapter_name(_start=[1]):
        name = str(_start[0])
        _start[0] += 1
        return name


class Adapter(object):
        """
        An adapter knows how to match itself to a read.
        In particular, it knows where it should be within the read and how to interpret
        wildcard characters.

        where --  One of the BACK, FRONT, PREFIX, SUFFIX or ANYWHERE constants.
                This influences where the adapter is allowed to appear within in the
                read and also which part of the read is removed.

        sequence -- The adapter sequence as string. Will be converted to uppercase.
                Also, Us will be converted to Ts.

        max_error_rate -- Maximum allowed error rate. The error rate is
                the number of errors in the alignment divided by the length
                of the part of the alignment that matches the adapter.

        minimum_overlap -- Minimum length of the part of the alignment
                that matches the adapter.

        read_wildcards -- Whether IUPAC wildcards in the read are allowed.

        adapter_wildcards -- Whether IUPAC wildcards in the adapter are
                allowed.

        name -- optional name of the adapter. If not provided, the name is set to a
                unique number.
        """
```

```
        def __init__(self, sequence, where, max_error_rate=0.1, min_overlap=3,
                        read_wildcards=False, adapter_wildcards=True, name=None, indels=True):
                self.debug = False
                self.name = _generate_adapter_name() if name is None else name
                self.sequence = parse_braces(sequence.upper().replace('U', 'T'))
                assert len(self.sequence) > 0
                self.where = where
                self.max_error_rate = max_error_rate
                self.min_overlap = min(min_overlap, len(self.sequence))
                self.indels = indels
                self.adapter_wildcards = adapter_wildcards and not set(self.sequence) <=
set('ACGT')
                self.read_wildcards = read_wildcards
                # redirect trimmed() to appropriate function depending on adapter type
                trimmers = {
                        FRONT: self._trimmed_front,
                        PREFIX: self._trimmed_front,
                        BACK: self._trimmed_back,
                        SUFFIX: self._trimmed_back,
                        ANYWHERE: self._trimmed_anywhere
                }
                self.trimmed = trimmers[where]
                if where == ANYWHERE:
                        self._front_flag = None  # means: guess
                else:
                        self._front_flag = where not in (BACK, SUFFIX)
                # statistics about length of removed sequences
                self.lengths_front = defaultdict(int)
                self.lengths_back = defaultdict(int)
                self.errors_front = defaultdict(lambda: defaultdict(int))
                self.errors_back = defaultdict(lambda: defaultdict(int))
                self.adjacent_bases = { 'A': 0, 'C': 0, 'G': 0, 'T': 0, '': 0 }

                self.aligner = align.Aligner(self.sequence, self.max_error_rate,
                        flags=self.where, wildcard_ref=self.adapter_wildcards,
wildcard_query=self.read_wildcards)
                self.aligner.min_overlap = self.min_overlap
                if not self.indels:
                        # TODO
                        # When indels are disallowed, an entirely different algorithm
                        # should be used.
                        self.aligner.indel_cost = 100000

        def __repr__(self):
                return '<Adapter(name="{name}", sequence="{sequence}", where={where}, '\
                        'max_error_rate={max_error_rate}, min_overlap={min_overlap}, '\
                        'read_wildcards={read_wildcards}, '\
                        'adapter_wildcards={adapter_wildcards}, '\
                        'indels={indels})>'.format(**vars(self))

        def enable_debug(self):
                """
                Print out the dynamic programming matrix after matching a read to an
                adapter.
                """
                self.debug = True
                self.aligner.enable_debug()

        def match_to(self, read):
                """
                Attempt to match this adapter to the given read.

                Return an Match instance if a match was found;
                return None if no match was found given the matching criteria (minimum
                overlap length, maximum error rate).
                """
                read_seq = read.sequence.upper()
                pos = -1
                # try to find an exact match first unless wildcards are allowed
                if not self.adapter_wildcards:
                        if self.where == PREFIX:
                                pos = 0 if read_seq.startswith(self.sequence) else -1
                        elif self.where == SUFFIX:
                                pos = (len(read_seq) - len(self.sequence)) if
read_seq.endswith(self.sequence) else -1
                        else:
                                pos = read_seq.find(self.sequence)
                if pos >= 0:
                        match = Match(
```

```
                                0, len(self.sequence), pos, pos + len(self.sequence),
                                len(self.sequence), 0, self._front_flag, self, read)
                else:
                        # try approximate matching
                        if not self.indels and self.where in (PREFIX, SUFFIX):
                                if self.where == PREFIX:
                                        alignment = align.compare_prefixes(self.sequence,
read_seq,
                                                wildcard_ref=self.adapter_wildcards,
wildcard_query=self.read_wildcards)
                                else:
                                        alignment = align.compare_suffixes(self.sequence,
read_seq,
                                                wildcard_ref=self.adapter_wildcards,
wildcard_query=self.read_wildcards)
                                astart, astop, rstart, rstop, matches, errors = alignment
                                if astop - astart >= self.min_overlap and errors / (astop -
astart) <= self.max_error_rate:
                                        match = Match(*(alignment + (self._front_flag, self,
read)))
                                else:
                                        match = None
                        else:
                                alignment = self.aligner.locate(read_seq)
                                if self.debug:
                                        print(self.aligner.dpmatrix)  # pragma: no cover
                                if alignment is None:
                                        match = None
                                else:
                                        astart, astop, rstart, rstop, matches, errors =
alignment
                                        match = Match(astart, astop, rstart, rstop, matches,
errors, self._front_flag, self, read)

                if match is None:
                        return None
                assert match.length > 0 and match.errors / match.length <=
self.max_error_rate, match
                assert match.length >= self.min_overlap
                return match

        def _trimmed_anywhere(self, match):
                """Return a trimmed read"""
                if match.front:
                        return self._trimmed_front(match)
                else:
                        return self._trimmed_back(match)

        def _trimmed_front(self, match):
                """Return a trimmed read"""
                # TODO move away
                self.lengths_front[match.rstop] += 1
                self.errors_front[match.rstop][match.errors] += 1
                return match.read[match.rstop:]

        def _trimmed_back(self, match):
                """Return a trimmed read without the 3' (back) adapter"""
                # TODO move away
                self.lengths_back[len(match.read) - match.rstart] += 1
                self.errors_back[len(match.read) - match.rstart][match.errors] += 1
                adjacent_base = match.read.sequence[match.rstart-1:match.rstart]
                if adjacent_base not in 'ACGT':
                        adjacent_base = ''
                self.adjacent_bases[adjacent_base] += 1
                return match.read[:match.rstart]

        def __len__(self):
                return len(self.sequence)


class ColorspaceAdapter(Adapter):
        def __init__(self, *args, **kwargs):
                super(ColorspaceAdapter, self).__init__(*args, **kwargs)
                has_nucleotide_seq = False
                if set(self.sequence) <= set('ACGT'):
                        # adapter was given in basespace
                        self.nucleotide_sequence = self.sequence
                        has_nucleotide_seq = True
                        self.sequence = colorspace.encode(self.sequence)[1:]
```

```
                if self.where in (PREFIX, FRONT) and not has_nucleotide_seq:
                        raise ValueError("A 5' colorspace adapter needs to be given in
nucleotide space")
                self.aligner.reference = self.sequence

        def match_to(self, read):
                """Return Match instance"""
                if self.where != PREFIX:
                        return super(ColorspaceAdapter, self).match_to(read)
                # create artificial adapter that includes a first color that encodes the
                # transition from primer base into adapter
                asequence = colorspace.ENCODE[read.primer + self.nucleotide_sequence[0:1]] +
self.sequence

                pos = 0 if read.sequence.startswith(asequence) else -1
                if pos >= 0:
                        match = Match(
                                0, len(asequence), pos, pos + len(asequence),
                                len(asequence), 0, self._front_flag, self, read)
                else:
                        # try approximate matching
                        self.aligner.reference = asequence
                        alignment = self.aligner.locate(read.sequence)
                        if self.debug:
                                print(self.aligner.dpmatrix)  # pragma: no cover
                        if alignment is not None:
                                match = Match(*(alignment + (self._front_flag, self, read)))
                        else:
                                match = None

                if match is None:
                        return None
                assert match.length > 0 and match.errors / match.length <= self.max_error_rate
                assert match.length >= self.min_overlap
                return match

        def _trimmed_front(self, match):
                """Return a trimmed read"""
                read = match.read
                self.lengths_front[match.rstop] += 1
                self.errors_front[match.rstop][match.errors] += 1
                # to remove a front adapter, we need to re-encode the first color following
the adapter match
                color_after_adapter = read.sequence[match.rstop:match.rstop + 1]
                if not color_after_adapter:
                        # the read is empty
                        return read[match.rstop:]
                base_after_adapter = colorspace.DECODE[self.nucleotide_sequence[-1:] +
color_after_adapter]
                new_first_color = colorspace.ENCODE[read.primer + base_after_adapter]
                new_read = read[:]
                new_read.sequence = new_first_color + read.sequence[(match.rstop + 1):]
                new_read.qualities = read.qualities[match.rstop:] if read.qualities else None
                return new_read

        def _trimmed_back(self, match):
                """Return a trimmed read"""
                # trim one more color if long enough
                adjusted_rstart = max(match.rstart - 1, 0)
                self.lengths_back[len(match.read) - adjusted_rstart] += 1
                self.errors_back[len(match.read) - adjusted_rstart][match.errors] += 1
                return match.read[:adjusted_rstart]

        def __repr__(self):
                return '<ColorspaceAdapter(sequence={0!r}, where={1})>'.format(self.sequence,
self.where)


class LinkedMatch(object):
        """
        Represent a match of a LinkedAdapter.

        TODO
        It shouldn't be necessary to have both a Match and a LinkedMatch class.
        """
        def __init__(self, front_match, back_match, adapter):
                self.front_match = front_match
                self.back_match = back_match
                self.adapter = adapter
```

```
                assert front_match is not None


class LinkedAdapter(object):
        """
        """
        def __init__(self, front_sequence, back_sequence,
                        front_anchored=True, back_anchored=False, name=None, **kwargs):
                """
                kwargs are passed on to individual Adapter constructors
                """
                assert front_anchored and not back_anchored
                where1 = PREFIX if front_anchored else FRONT
                where2 = SUFFIX if back_anchored else BACK
                self.front_anchored = front_anchored
                self.back_anchored = back_anchored

                # The following attributes are needed for the report
                self.where = LINKED
                self.name = _generate_adapter_name() if name is None else name
                self.front_adapter = Adapter(front_sequence, where=where1, name=None,
**kwargs)
                self.back_adapter = Adapter(back_sequence, where=where2, name=None, **kwargs)

        def enable_debug(self):
                self.front_adapter.enable_debug()
                self.back_adapter.enable_debug()

        def match_to(self, read):
                """
                Match the linked adapters against the given read. If the 'front' adapter
                is not found, the 'back' adapter is not searched for.
                """
                front_match = self.front_adapter.match_to(read)
                if front_match is None:
                        return None
                # TODO use match.trimmed() instead as soon as that does not update
                # statistics anymore
                read = read[front_match.rstop:]
                back_match = self.back_adapter.match_to(read)
                return LinkedMatch(front_match, back_match, self)

        def trimmed(self, match):
                front_trimmed = self.front_adapter.trimmed(match.front_match)
                if match.back_match:
                        return self.back_adapter.trimmed(match.back_match)
                else:
                        return front_trimmed

        # Lots of forwarders (needed for the report). I'm sure this can be done
        # in a better way.

        @property
        def lengths_front(self):
                return self.front_adapter.lengths_front

        @property
        def lengths_back(self):
                return self.back_adapter.lengths_back

        @property
        def errors_front(self):
                return self.front_adapter.errors_front

        @property
        def errors_back(self):
                return self.back_adapter.errors_back

        @property
        def adjacent_bases(self):
                return self.back_adapter.adjacent_bases
```

Appendix VII: Optional method nr 4

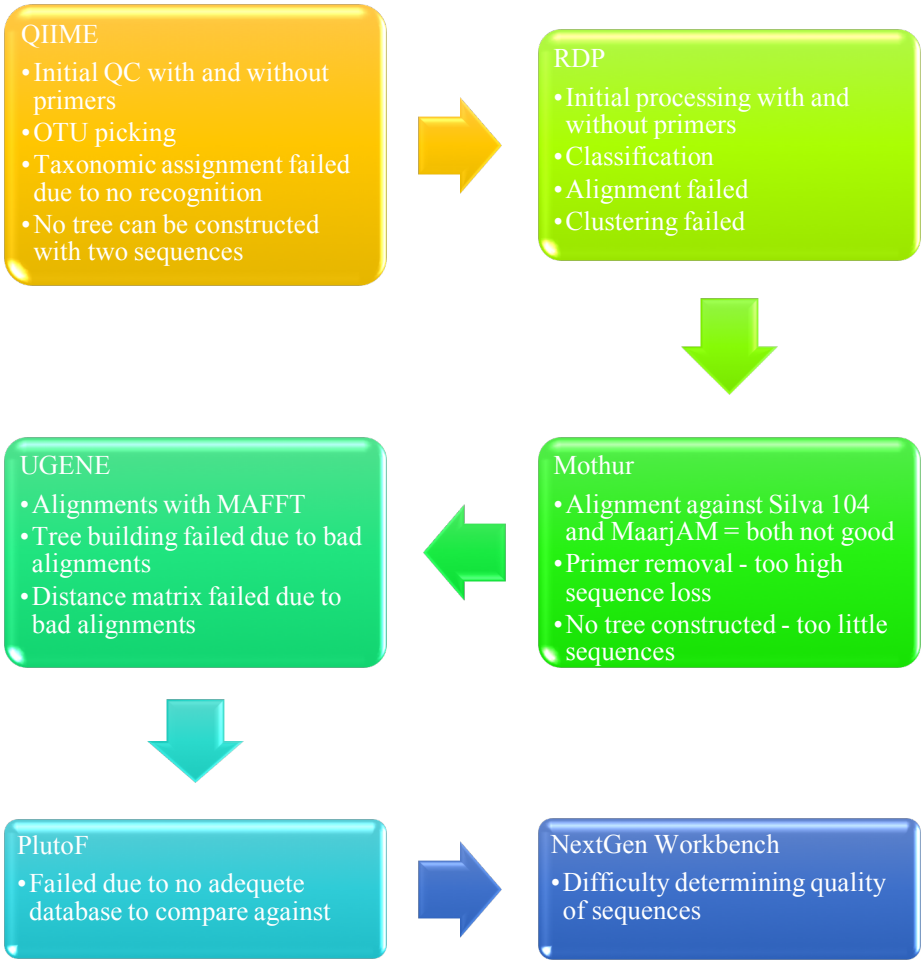


Figure VII.1: Illustration and summary of the pipelines and standalone programs tried in order to analyse the 18S AM fungal data. The headers in each section refers to the pipeline or standalone program that was used. The points beneath the header stated the main reasons why the specific pipeline or program was not successful in analyzing the 18S AM fungal data.

UGENE

After seeing the results for all of the samples in all the above pipelines, UGENE was implemented to do the alignments. T-COFFEE was the alignment algorithm chosen originally, because this method was a consistency-based MSA tool that attempts to mitigate the pitfalls of progressive alignment methods and is good to use on small sample files. With T-COFFEE the gap opening penalty was left as the default as -50 and the gap extension penalty was left as 0. But T-COFFEE did fail and so MAFFT was run. MAFFT was chosen because it can be used with sequences that were from a medium to large size.

Chimera removal

The chimera checking tool on the FunGenePipeline uses the tool UCHIME (Edgar *et al*., 2011) that uses USEARCH 6.0 and is limited to 60,000 sequences through the website, but this

limitation may be increased when using the USEARCH website. USEARCH was run in UCHIME *de novo* mode which meant that the tool would only be able to work with experimental data that contain both chimeric and non-chimeric sequences at higher abundances (Fish *et al*., 2013).

After chimera removal with USEARCH 6.0 in order to check whether the amplicon sequence files contained any chimeras, various different folders were the output: **chimera_filtered_sequences**, **filtered_mapping** and **chimera_check**. But the file used to proceed with clustering was the *all_seqs_derep.fasta* file which was just edited to contain the tag in the filename.

RDP classifier

The RDP pipeline classifier was used to assign sequences that were derived from fungal genes to the corresponding taxonomic model. The classifier is limited to 500,000 sequences to be analysed. The UNITE fungal ITS trainset 1 from 2014 was used as the model, for interest sake, to be able to have a comparison of results, and this was also the only available fungal model with the classifier tool. The input can be single or multiple FASTA, FASTQ or SFF files, but it requires at least 50 bases in order to get a good classification result. The output will be *bootstrap_conf.txt*, *classification.txt*, *hierarchy.txt* and *failed_sequences.txt* files.

The *bootstrap_conf.txt* contained the average bootstrap scores for each rank. The *classifications.txt* contained a sequence-by-sequence classification result that included all the confidence scores at each level of the hierarchy. The *hierarchy.txt* file contained the sequence counts for each taxon in the hierarchy which could be imported into RStudio for analysis using the package phyloseq. The *failed_sequences.txt* file contained all the sequences that failed during classification.

In statistics, bootstrapping can refer to any test or metric that relies on random sampling with replacement. Bootstrapping allowed assigning measures of accuracy (defined in terms of bias, variance, confidence intervals, prediction error or some other such measure) to the sample estimates.

Analysis of results

In RStudio, the Bioconductor packages MetagenomeSeq and PhyloSeq can be downloaded. These packages can also be used later on with other types of analysis, for example after OTU picking and tree generation in the QIIME pipeline. Analysis can be done on the hierarchy.txt file generated by the RDP classifier in order to show the sequence counts for each taxon in the hierarchy.

According to the BioStars website (https://www.biostars.org) there is a website calles T-Rex that can also be used to construct phylogenetic trees and this webserver is also much more straight forward than using MEGA, but this may lead to the loss of some quality. No methods were found to draw a phylogenetic tree in RStudio from the classifier data generated by the RDP pipeline.

Using BLAST and MaarjAM

All the sample files were BLASTed against the MaarjAM database to see what results could be found. Problems earlier encountered in the QIIME pipeline was that the VT keys were not recognizable as in the case of that found on the more known databases such as Silva 104 and GenBank. These BLAST results could then help to make alignment files which could be used to draw phylogenetic trees and do further downstream analysis on. Possibly cluster analysis on the RDP pipeline can also be done and heatmaps drawn to show the diversity of the samples. But this also lead to problems. After BLASTing the samples against the MaarjAM database, the masked text file (chapter 3) was used as the input for the RDP pipeline in order to do complete linkage clustering analysis. The problem came in when the RDP pipeline did not recognize any of the MaarjAM VT keys and do not have keys/identifiers that agree with that used in more common databases such as GenBank.

Using R and RStudio

NCBI was searched for “arbuscular mycorrhizal fungi 18S” and yielded 40 394 results of various different AM fungi that were 18S DNA/RNA sequences. When loading into RStudio, only 11 962 were in the AM fungi value. The following information was adapted from the Bioinformatics 0.1 documentation compiled by Avril Coghlan:

Using the SeqinR package in RStudio, DNA reads in a FASTA file can easily be read into R. If the FASTA sequence file is not available, the getncbiseq() function can be used and this

sequence can then also be saved in the FASTA format. The function read.fasta() was used to read the FASTA file into R. The working directory also needed to be set correctly before starting to work with the setwd() function. The read.fasta() function reads the FASTA file into an object that is listed in R as a variable. A list in R is an object that has properties of a vector but it can also contain characters. In this case the FASTA files that were read in contained information regarding the sequence as DNA. The elements of an R list object can then be accessed using double square brackets and the variable will be stored as a vector that contains the nucleotide sequences.

Once the DNA sequence was stored as a vector, some basic statistics was done in order to describe the different sequences, for example the total length of the sequences in nucleotides. In order to obtain the length of the genome sequence, the length() function was used which then gave the amount of sequences that were present in the vectors. When using FASTA files containing only one sequence, the length() function will return the length of the sequence stored in the specific variable in nucleotides. This function gives the number of elements in the input vector that was passed to it.

Another basic first analysis method of the sequences were to count the number of occurrences of the four different nucleotides present in the sequences. This was done by using the table() function. This function determined the average count of each nucleotide per element in the sequence files.

Another basic statistic that was done on the sequences was to calculate the GC content of the sequences in each file. The GC content can simply be defined as the fraction of the sequences that consist of Gs or Cs ie. %(Gs+Cs).

$$GC\ content = \frac{(number\ of\ Gs + number\ of\ Cs) * 100}{genome\ length}$$

Alternatively, the function GC() was used to calculate the average GC content of each sequence in the files. This function returned the fraction of bases in the sequences that were Gs or Cs. This fraction was easily converted to percentage by multiplying it by 100.

It was also interesting to determine what “DNA words” were present in the sequences. The average frequency of “DNA words” that were 3 bases long were determined. This helped in determining the presence of the start and stop codons in the sequences after which it could be said that if both start and stop codons were present in most of the sequences, then the whole 18S gene was amplified. If the whole 18S gene was not amplified, then this could have an influence on the analysis and alignments against 18S sequences in Silva 104 and MaarjAM that were used as reference databases. This can also explain why the alignments for the Rooibos natural and commercial samples against the respective reference databases did not have good results. In order to determine the presence of the 3 letter “words”, the function *count(vector,3)* was used, where the vector is the vector previously created that contained all the sequences as elements.

Another way that could have been used to determine the presence of the start and stop codons could have been to use the function count(vector,2) which would have included all overlapping DNA words present in the sequence. It can be used by saying, for example, the start codon “ATG” will be considered to contain two “words” that are each two nucleotides long namely “AT” and “GC”. If the count() function was used by itself, the the resulting output would have been a table object. Double square brackets could then be used to extract certain values of the elements from the table. In other cases the value of the element in the table can just be typed in.

Table VII.1: Summary of the files used in RStudio. This included files after loading all the files into R and doing basic statistical analysis.

Sample/Reference in FASTA format	Number of elements present	Average number of bases per sequence in each sample/reference				GC content (%)	Start codon present (ATG)	Stop codons present (TGA, TAA, TAG)
		A	C	G	T			
AMF_NCBI	11962	132	79	86	162	35.95	Yes	Yes
MaarjAM_vt_2015	352	130	103	141	142	47.29	Yes	Yes
Honeybush natural	15680	18	25	32	23	58.16	Yes	Yes
Honeybush commercial	10165	6	5	15	9	57.14	Yes	Yes
Rooibos natural	16498	22	16	19	16	47.95	Yes	No
Rooibos commercial	4437	11	16	5	9	51.22	No	No

What the above table showed is that there might be some reason as to why some of the pipelines did not work as well as expected. Not all the start and stop codons were present in the sequences and this might have had an influence on whether the entire 18S gene was amplified or not in the specific samples. Futher investigation has to be done in this regard to determine whether this might be a reason for poor results in the pipeline analysis.

****