DArTseq_pipeline Manual

by **Sander de Backer**
Crop Wild Relatives
Meise Botanic Garden

2022

Contents

1 Diversity Arrays Technology Pipeline

This document explains the DArTseq technique and general workflow to process raw fastq reads generated by the DArTseq platform (Diversity Arrays Technology, Canberra, Australia), as well as multiple other analysis protocols for phylogenetics, metagenomics, population genetics, and hybridization. Additional information about techniques, tools, workflows, etc. can always be found in the references mentioned at the end of each section. These are either published papers, online manuals, or repositories for the topics in question.

1.1 DArT Sequencing

DArT is initiated by defining a gene pool representing the germplasm to be analysed. DArT utilises restriction enzymes (REs) to achieve genomic complexity reduction as REs offer a high level of precision (selectivity and reproducibility) [1]. The enzymes selected may differ depending on the genus/species of interest. For *Musa* spp., PstI and MseI are used (important for the data analysis). The digestion fragments are cloned and amplified before hybridizing targets (fluorescently labelled genomic representations) from a specific sample to an array containing a large collection of probes amplified from bacterial clones from a representation of the gene pool of interest (**Figure 1**). For detailed assessment of DArT on *Musa* spp., check Risterucci *et al.* (2009) [2].

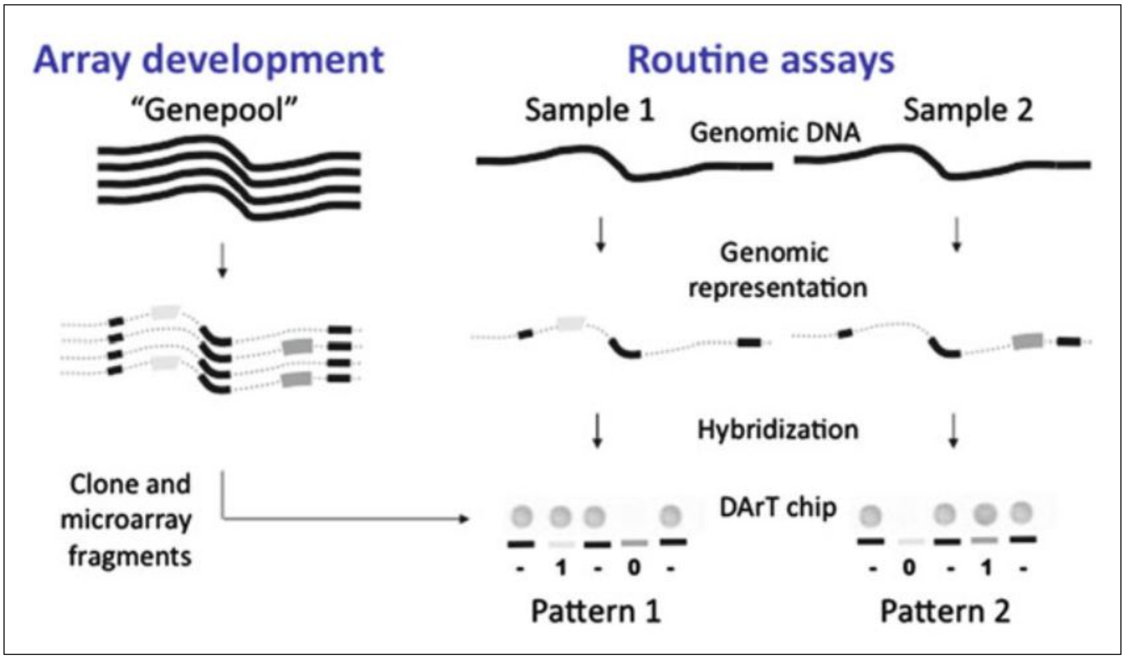
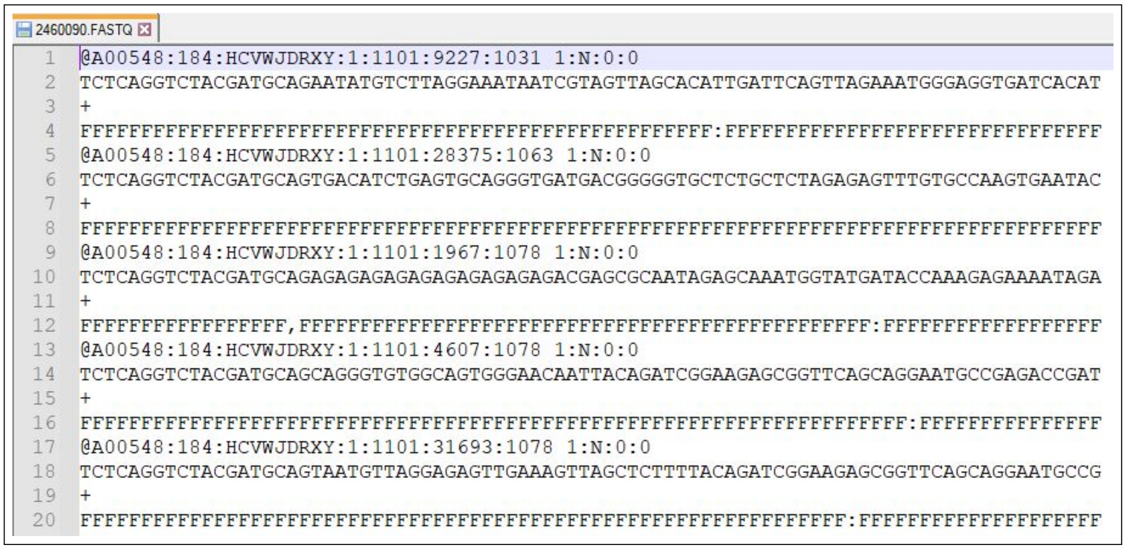


Figure 1: Principles of a DArT analysis. Genomic complexity reduction with restriction enzymes, and hybridisation between genepool and samples on the DArT array.

DArTseq is an effort in "genotyping by sequencing" the complexity-reduced fractions of plant genomes [3]. Sequencing is performed on an Illumina platform and generates stacked short read data. The choice of service (low, medium, high density) affects both the read depth and the read density. As DArT uses methylation-sensitive REs (like PstI) for making the libraries, the DArTseq markers are mainly located in active regions and far less

abundant in the centromeric, methylated regions, therefore avoiding the predominantly repetitive regions of the plant genome. More information on the Illumina sequencing technology can be found here:
`https://www.illumina.com/documents/products/techspotlights/techspotlight_sequencing.pdf`
`https://www.illumina.com/science/technology/next-generation-sequencing/sequencing-technology.html`
Raw data are generated in FASTQ format. These files can be opened with text editors like Notepad++. Each individual read is represented by four lines. Line1 begins with the '@' character and contains information about the instrument, flowcell, run ID, cluster, sample index ... Line2 is the raw sequence of nucleotides. Line3 begins with the '+' character and optionally follows-up with identifiers as found in Line1. Line4 encodes the quality values of the sequence in Line2. Each character corresponds with one nucleotide. For example, 'F' means a score of 37 out of 40 (**Figure 2**).



```
@A00548:184:HCVWJDRXY:1:1101:9227:1031 1:N:0:0
TCTCAGGTCTACGATGCAGAATATGTCTTAGGAAATAATCGTAGTTAGCACATTGATTCAGTTAGAAATGGGAGGTGATCACAT
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@A00548:184:HCVWJDRXY:1:1101:28375:1063 1:N:0:0
TCTCAGGTCTACGATGCAGTGACATCTGAGTGCAGGGTGATGACGGGGGTGCTCTGCTCTAGAGAGTTTGTGCCAAGTGAATAC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@A00548:184:HCVWJDRXY:1:1101:1967:1078 1:N:0:0
TCTCAGGTCTACGATGCAGAGAGAGAGAGAGAGAGAGAGACGAGCGCAATAGAGCAAATGGTATGATACCAAAGAGAAAATAGA
+
FFFFFFFFFFFFFFFFF,FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFF
@A00548:184:HCVWJDRXY:1:1101:4607:1078 1:N:0:0
TCTCAGGTCTACGATGCAGCAGGGTGTGGCAGTGGGAACAATTACAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGAT
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFF
@A00548:184:HCVWJDRXY:1:1101:31693:1078 1:N:0:0
TCTCAGGTCTACGATGCAGTAATGTTAGGAGAGTTGAAAGTTAGCTCTTTTACAGATCGGAAGAGCGGTTCAGCAGGAATGCCG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFF
```

Figure 2: FASTQ file formatting as visualised in Notepad++.

1.2 DArTseq Analysis Pipeline

The processing and analysis pipeline of raw fastq read data uses the UNIX environment (e.g. Linux, Ubuntu, OSX...). Systems running OSX require different installation commands which are not presented in this document. Explained below is the installation of Ubuntu for computers running the Windows operating system, and the required packages/tools for the analysis pipeline. Be aware that every installation can vary in settings, however small the difference. Read every warning and error carefully, most are resolved easily and quickly, and have probably already been solved by the tool's community. Also remember that Google and StackOverflow are your friends!

1.2.1 Ubuntu for Windows

On Windows 10, follow these steps to install Ubuntu: (1) Go to Settings > Update & Security > For Developers > turn 'Developer Mode' to 'ON'. This enables the Windows Subsystem for Linux. (2) Go to Control Panel > Programs and Features > Turn Windows feature on or off > enable 'Windows Subsystem for Linux' and press

'OK'. (3) For these changes to take effect, reboot the computer. (4) From the Windows Store install 'Ubuntu'. (5) Run the Ubuntu application and follow the instructions shown in the terminal to complete the Ubuntu kernel installation within Windows. Choose username and password carefully, they will be used frequently hereafter. Keep Ubuntu up to date with commands below. Lines beginning with '#' contain only information. Each `sudo` command will ask for the user password, unless carried out in a certain timeframe.

Ubuntu terminal

```
# Search for available updates
sudo apt update
# Download and install updates
sudo apt upgrade
# First time setup, essential packages
sudo apt-get install build-essential
```

1.2.2 Dependencies and Tools

These packages can be installed in the home directory unless specified otherwise. Simply run the commands as shown below. Packages can be installed with `sudo` or `pip`. The `python3-pip` package enables installation with pip.

```
# Command to return to home directory (from wherever)
# cd = Change Directory
cd
# Installations with sudo
sudo apt install python3 python3-pip python-is-python3
sudo apt install fastqc
sudo apt install bwa
# Installations with pip
pip install multiqc
pip install cutadapt==3.5 # specifies that we want version 3.5
pip install gbprocess-ngs # should be 4.0.0.post1
```

For bcftools and samtools [4], go to `http://www.htslib.org/download/` and download the latest version of bcftools, samtools, and htslib. First, install htslib with the commands below. The other two packages are installed in exactly the same way, using their file names instead. Define `USER` with the chosen username upon installing Ubuntu. Also replace the `1.x` in the filename with the downloaded version.

```
# Install dependencies
sudo apt install libbz2-dev liblzma-dev libncurses5-dev zlib1g-dev
# Similarly for bcftools-1.x and samtools-1.x
# Move the downloaded file into the Ubuntu home directory
cd
mv /mnt/c/Users/USER/Downloads/htslib-1.x.tar.bz2 .
# Unpack archive and go into the folder
tar -xvf htslib-1.x.tar.bz2
```

```
rm htslib-1.x.tar.bz2
cd htslib-1.x
# Configure and install
./configure --prefix=/home/USER/htslib-1.x
make
make install
# ONLY FOR HTSLIB
# add line to .bashrc
export HTSLIB="/home/USER/htslib-1.15.1"
```

The latest FreeBayes [5] release is available on the GitHub repository: `https://github.com/freebayes/freebayes`. The latest static version for Linux can be downloaded from the 'Releases' page and simply be extracted and placed in a location accessible by $PATH.

```
# Download latest static version
wget https://github.com/freebayes/freebayes/releases/tag/
      v1.3.6/freebayes-1.3.6-linux-amd64-static.gz
# Unpack
gunzip freebayes-1.3.6-linux-amd64-static.gz
# Move pre-built static and rename to freebayes
sudo mv freebayes-1.3.6-linux-amd64-static /usr/bin/freebayes
```

The installed packages can simply be executed with their basic command from anywhere if their installation directory is part of the `\$PATH`. This can be checked by a command or looking into the .bashrc file in the home directory of the Ubuntu terminal. Remember to fill in your username!

```
# Command to show $PATH
echo $PATH
# Edit .bashrc from home directory
nano .bashrc
# Edit .bashrc from anywhere else
nano $HOME/.bashrc
# Standard example of $PATH in .bashrc
export PATH="$PATH:$HOME/bin:/usr/bin"
# With additional path added
export PATH="$PATH:$HOME/bin:/usr/bin:$HOME/.local/bin"
```

Picard: go to `https://broadinstitute.github.io/picard/` [6] and download the 'Latest Release' (file is always named 'picard.jar' regardless of the version). Place this .jar file in a directory easily accessible (and remember the path to the file!).

GATK4: go to `https://gatk.broadinstitute.org/hc/en-us` and 'Download the latest version of GATK' (GATK4.2) [7]. Extract this .zip file and find inside the folder a `gatk` file with no extension (this is the one we need). Rename the folder to GATK4 and remember the path to the `gatk` file!

1.2.3 Reference Preparation

The DArTseq_pipeline will process FASTQ.gz files into Variant Call Format (VCF) files. For the pipeline to run smoothly, follow this setup. Make a folder 'Reference' for the references used in the analyses. The mapping tool `bwa mem` and the AddOrReplaceReadGroups function of Picard both need additional reference index files. They are generated with the commands below. Replace `REF.fasta` with the name of the reference you want to use, and specify the path to the Picard file (picard.jar).

```
# Index for BWA MEM mapping
bwa index REF.fasta
# Index for SAM sorting with Picard
samtools faidx REF.fasta
java -jar picard.jar CreateSequenceDictionary -R REF.fasta
```

1.2.4 GBprocesS Configuration

The GBprocesS step requires a configuration file [8]. The scripts and other files used in this pipeline (and further analyses) can be downloaded from the GitHub repository:
`https://github.com/sanderdebacker/DArTseq_pipeline`
Discussed below are the different modules defining the operations in the GBprocesS_SE.ini file.

General The configuration of basic settings. The number of cores (threads) for the analysis with `cores` (default = 1). The input directory for gathering FASTQ(.gz) files is by default the directory where the GBprocesS command is carried out, but can be specified if needed. The `sequencing_type` can be either `se` (single-end) or `pe` (paired-end). Every input FASTQ(.gz) file is required to have a filename of the same length. The `{sample_name:}` specifies the name length without extensions, while the `{extension:}` specifies the extension length (FASTQ = 5, the bgzipped .gz extension is not taken into account!). To prevent flooding of the directory of execution with temporary files, a temporary folder can be given to work with (GBprocesS doesn't create this, need to be done manually).

Configuration file

```
[General]
cores = 2
input_directory =
sequencing_type = se
input_file_name_template = {sample_name:39}{extension:5}
temp_dir = temp/
```

CutadaptTrimmer This block describes the barcodes, restriction site remnants, and sequencing primers used in the NGS analysis. DArTseq uses the Illumina platform, therefore the reverse compliment of the universal Illumina adapter (AGATCGGAAGAGC) is filled in as `common_side_sequencing_primer` (= GCTCTTCCGATCT). The cutsite remnants match those of PstI and MseI (used in DArTseq analyses for *Musa* spp.), CTGCA and TTAN respectively.

GBprocesS uses an input fasta file containing all the barcodes of the samples to process. These barcodes can be gathered with the Identufy_barcodes_SdB.py script from:
`https://github.com/sanderdebacker/DArTseq_pipeline`

It is a modified script extracting the barcodes of all FASTQ(.gz) files in the directory of execution, and writing the TableOfBarcodes.fasta file. This script is not flawless, check the fasta file to ensure every sample has a barcode before further analysis. Add missing barcodes manually by comparing a couple of reads from the same sample. The adapters for DArTseq analysis are not anchored (`False`). The required minimum length of reads after trimming is 25 nucleotides, otherwise the reads are discarded. The allowed error rate in barcodes is 0.1. The output directory is by default the directory of execution and an output filename can be specified.

```
[CutadaptTrimmer]
common_side_sequencing_primer = GCTCTTCCGATCT
barcode_side_cutsite_remnant = CTGCA
common_side_cutsite_remnant = TTAN
barcodes = TableOfBarcodes.fasta
anchored_adapters = False
minimum_length = 25
error_rate = 0.1
output_directory =
output_file_name_template = {sample_name}.Trimmed{extension}
```

MaxNFilter This block filters through the reads and removes those that have more than `max_n` undetermined nucleotides (N), writing new files with the provided `output_file_name_template`.

```
[MaxNFilter]
max_n = 5
output_directory =
output_file_name_template = {sample_name}.Trimmed.Max5N{extension}
```

AverageQualityFilter The average quality of a read has to be equal to or greater than `average_quality`, removing those that fail this criteria. The ASCII quality score line of the .fastq records are interpreted using the Illumina 1.8+ Phred+33 encoding (**Figure 2**).

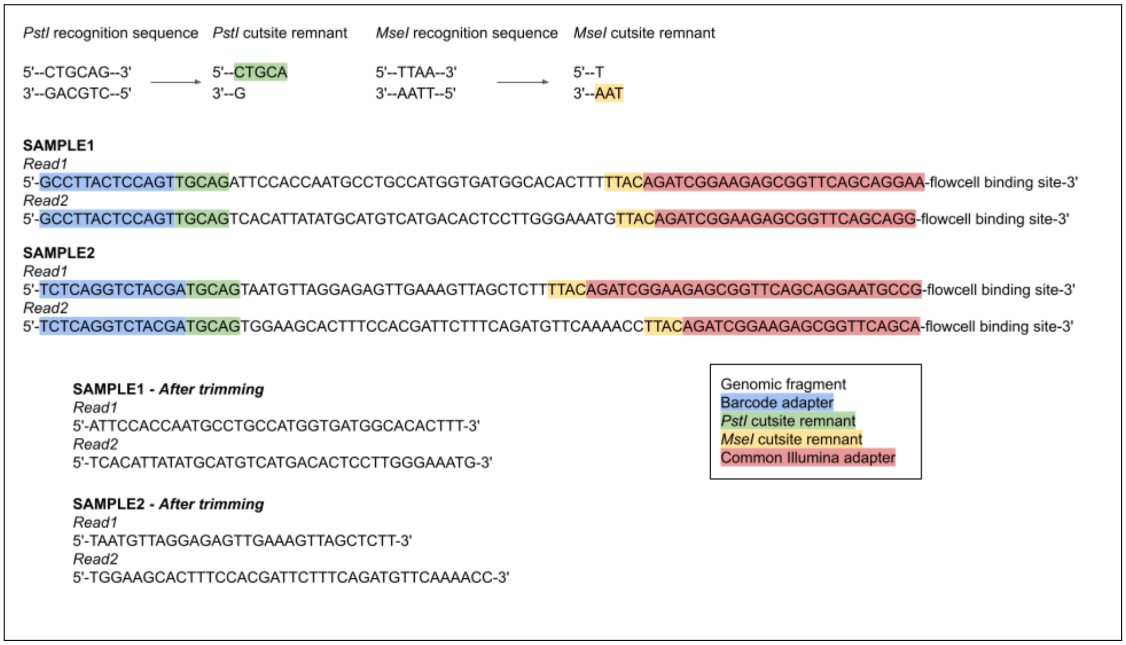
```
[AverageQualityFilter]
average_quality = 25
output_directory =
output_file_name_template = {sample_name}.Trimmed.Max5N.Avq25{extension}
```

RemovePatternFilter Reads that contain intact restriction enzyme sites for both enzymes used (PstI and MseI) have to be removed. The output of this last step is a cleaned FASTQ file with extension `.GBprocesS.fastq.gz` (stated in the `output_file_name_template` option), recognized as input for the next step in the pipeline.

```
#Remove reads with intact PstI restriction site.
[RemovePatternFilter.PstI]
```

```
pattern = CTGCAG
output_directory =
output_file_name_template = {sample_name}.Trimmed.Max5N.Avq25.woPstIsites{extension}

#Remove reads with intact MseI restriction site.
[RemovePatternFilter.MseI]
pattern = TTAA
output_directory =
output_file_name_template = {sample_name}.GBprocesS{extension}
```



PstI recognition sequence
PstI cutsite remnant
MseI recognition sequence
MseI cutsite remnant
5'--CTGCAG--3'
3'--GACGTC--5'
5'--CTGCA
3'--G
5'--TTAA--3'
3'--AATT--5'
5'--T
3'--AAT
SAMPLE1
Read1
5'-GCCTTACTCCAGTTGCAGATTCCACCAATGCCTGCCATGGTGATGGCACACTTTTTACAGATCGGAAGAGCGGTTCAGCAGGAA-flowcell binding site-3'
Read2
5'-GCCTTACTCCAGTTGCAGTCACATTATATGCATGTCATGACACTCCTTGGGAAATGTTACAGATCGGAAGAGCGGTTCAGCAGG-flowcell binding site-3'
SAMPLE2
Read1
5'-TCTCAGGTCTACGATGCAGTAATGTTAGGAGAGTTGAAAGTTAGCTCTTTTACAGATCGGAAGAGCGGTTCAGCAGGAATGCCG-flowcell binding site-3'
Read2
5'-TCTCAGGTCTACGATGCAGTGGAAGCACTTTCCACGATTCTTTCAGATGTTCAAAACCTTACAGATCGGAAGAGCGGTTCAGCA-flowcell binding site-3'
SAMPLE1 - *After trimming*
Read1
5'-ATTCCACCAATGCCTGCCATGGTGATGGCACACTTT-3'
Read2
5'-TCACATTATATGCATGTCATGACACTCCTTGGGAAATG-3'
SAMPLE2 - *After trimming*
Read1
5'-TAATGTTAGGAGAGTTGAAAGTTAGCTCTT-3'
Read2
5'-TGGAAGCACTTTCCACGATTCTTTCAGATGTTCAAAACC-3'
Genomic fragment
Barcode adapter
PstI cutsite remnant
MseI cutsite remnant
Common Illumina adapter

Figure 3: Image for double-digest single-end sequencing (DArTseq with PstI and MseI for *Musa* spp.). Barcodes are the same for all reads within a sample, but differ between samples. More information available in the manual of GBprocesS: `https://gbprocess.readthedocs.io/en/latest/index.html`.

Before running the DArTseq_pipeline, the Gbprocess_SE.ini file for GBprocesS needs to be configured. The configuration needed depends on the analysis type, in this case for an analysis on Single End (SE) data. Also the filename and extension length matter! The extension consists of five characters (FASTQ; .gz does not count). Temporary files can be put in the folder 'temp/'. The reverse complement of the universal Illumina adapter (AGATCGGAAGAGC) is the common_side_sequencing_primer, with the restriction remnants of PstI (CTGCA) and MseI (TTAN)(**Figure 3**). Barcodes can be found in the TableOfBarcodes.fasta file generated with the Identify_barcodes_loop script. Replace the configuration arguments based on your analysis. For more information about GBprocesS, and examples of configuration (.ini) files for different analyses: `https://gbprocess.`

`readthedocs.io/en/latest/`.

1.2.5 DArTseq Analysis

The pipeline uses, in order, GBprocesS (with Cutadapt) for preprocessing of the reads, BWA MEM for mapping of the reads on the reference genome, picard AddOrReplaceReadGroups for sorting the mapped reads, FreeBayes to call variants (SNPs, MNPs, indels), and GATK SelectVariants to filter the VCF to SNP-only. The output VCF can be further filtered with GATK, vcftools, or bcftools based on your preference or requirements for further analysis.

Comments

Alternative At the GitHub repository `https://github.com/CathyBreton/Genomic\_Evolution`, an alternative pipeline can be found. It uses a Perl script with the GATK HaplotypeCaller, but is much more strict and less effective on stacked data (generated with DArTseq), as mentioned on the developers page: `https://bit.ly/3MOoq8z`

Picard Error If the `picard.jar AddOrReplaceReadGroups` step gives an error like `htsjdk.samtools.SAMLineParser()`, there could be problems with the header of the .sam file. The `picard.jar ReplaceSamHeader` can help to replace the wrong header with one of a file already analysed that didn't give an error.

2 Variant Call Format

The Variant Call Format (VCF) file stores genetic sequence variation in text file format [9]. In case of this manual representing the Single Nucleotide Polymorphisms (SNPs) found in the mapped reads on the reference genome. It also contains meta-information lines useful in filtering the SNPs and fine tuning further analyses.

2.1 VCF Statistics

Many options are available for a large variety of statistics. Mainly BCFtools [4] and VCFtools [9] are used for statistical output. These tools can generate statistics for single samples, but most are more interesting in relation to other/all samples in a data set. For this, the single sample VCF files need to be combined into a multi-sample VCF file. On how to merge VCF files, check **3.1 SNP Filtering** where it is explained in detail. For each tool, the main statistical output options are explained as also available on their respectful manual pages.

vctools (`https://vcftools.github.io/man_latest.html`)
`-het` Calculates a measure of heterozygosity on a per-individual basis. Specifically, the inbreeding coefficient, F, is estimated for each individual using a method of moments. The resulting file has the suffix `.het`.
`-hardy` Reports a p-value for each site from a Hardy-Weinberg Equilibrium test (as defined by Wigginton, Cutler and Abecasis (2005)). The resulting file (with suffix `.hwe`) also contains the Observed numbers of Homozygotes and Heterozygotes and the corresponding Expected numbers under HWE.
`-site-pi` Measures nucleotide divergence on a per-site basis. The output file has the suffix `.sites.pi`.
`-TsTv-summary` Calculates a simple summary of all Transitions and Transversions. The output file has the suffix `.TsTv.summary`.
`-geno-r2` Calculates the squared correlation coefficient between genotypes encoded as 0, 1 and 2 to represent the number of non-reference alleles in each individual. This is the same as the LD measure reported by PLINK. The D and D' statistics are only available for phased genotypes. The output file has the suffix `.geno.ld`.

bcftools (`http://samtools.github.io/bcftools/bcftools.html`)
`bcftools stats` Parses VCF and produces text file stats which is suitable for machine processing and can be plotted using plot-vcfstats. By default only sites are compared. When one VCF file is specified on the command line, then stats by non-reference allele frequency, depth distribution, stats by quality and per-sample counts, singleton stats, etc. are printed. When two VCF files are given, then stats such as concordance (Genotype concordance by non-reference allele frequency, Genotype concordance by sample, Non-Reference Discordance) and correlation are also printed. Per-site discordance (PSD) is also printed in `-verbose` mode.

Hierfstat (Rstudio package) [10] Estimates hierarchical F-statistics from haploid or diploid genetic data (HO, HS, FIS, FST, etc.).

2.2 VCF Data Visualisation

There are many ways to visualise the data compacted into a VCF file, individual or multi-sample. Below are only a few examples. Analyses are performed with different packages for RStudio.

2.2.1 Sample-Variant Matrix

With vcfR, a matrix can be created to visualise the variants that are present in each sample [11]. This also allocates a colour code to the sample-variant combination, highlighting the quality and probability of the SNP. Below is the Rscript used to generate the matrix (col=samples; row=variants). Important to note is the renaming of the rows (variants) to numeric values. This makes it possible to plot only a subsection of the total SNPs in the last line (example shows variant 1001 to variant 1500). The matrix quickly gets too crowded, therefore only visualise the range of variants necessary. The `masker()` function can be used to filter based on read depth (min_DP, max_DP) and quality (min_MQ, max_MQ).

RStudio script

```
# Install and load vcfR
install.packages('vcfR')
library(vcfR)
# Load VCF and reference
vcf <- read.vcfR("VCF.vcf.gz", verbose = TRUE )
dna <- ape::read.dna("REF.fasta", format = "fasta")
# Create chromR object
chrom <- create.chromR(name="Organism_Name", vcf=vcf, seq=dna, verbose=TRUE)
chrom <- masker(chrom, min_DP = 10, min_MQ = 20)
chrom <- proc.chromR(chrom, verbose = TRUE)
dp <- extract.gt(chrom, element="DP", as.numeric=TRUE)
# Optionally rename variants to numbers
rownames(dp) <- 1:nrow(dp)
# Create heatmap of all SNPs
heatmap.bp(dp)
# Generate PNG heatmap of SNP 1001-1500
png(filename="VCF_SNPs.png", height=5000, width=5000, res=300)
heatmap.bp(dp[1001:1500,])
dev.off()
```

Also a subdivision of the reference genome can be selected to be plotted. For this, the names of the reference chromosomes, contigs, or scaffolds need to be known. Only one command needs to be added, shown below in between two commands identical to the ones above. Here, only the SNPs of chromosome 01 are plotted against the samples in the data set.

```
dna <- ape::read.dna("REF.fasta", format = "fasta")
dna <- dna[grep("chr01", names(dna))]
chrom <- create.chromR(name="Organism_Name", vcf=vcf, seq=dna, verbose=TRUE)
```

2.2.2 Variant Correlations

SNPs that are close together are more likely to be inherited together. This correlation can be measured and visualised. Either for all variants, or a defined subset. First, convert the VCF file to `.csv` and delete all the rows above the header (rows commented out with #). This `.csv` will be the first input into RStudio.

```
# Install and load packages
install.packages("tidyverse")
library(tidyverse)
```

```
# Load VCF.csv and extract information
vcf <- read.csv("VCF.csv", sep = "\t")
row.names(vcf) <- paste(vcf$X.CHROM,vcf$POS,sep=" ")
vcf <- vcf[-c(1:9)]
vcf[] <- lapply(vcf[], function(x) substr(x, 1, 3))
vcf[vcf== "1/1"] <- "2"
vcf[vcf== "0/1"] <- "1"
vcf[vcf== "0/0"] <- "0"
vcf[vcf== "./."] <- NA
# Create correlation matrix
matrix <- data.matrix(vcf)
matrix <- t(matrix)
cor <- cor(matrix)
col <- colorRampPalette(c("blue", "white", "red"))(20)
# Generate PNG plot
png(filename="Variant_Correlation.png", height=5000, width=5000, res=300)
heatmap(cor, col = col, symm = TRUE, cexRow = 0.25, cexCol = 0.25)
dev.off()
```

This can also be shown with the Corrplot package for Rstudio [12]. The start is the same as above, only the heatmap function changes to a corrplot function. The corrplot type can be upper, lower, or full and `diag=FALSE` can be added to remove the diagonal.

```
# Install and load corrplot
install.packages("corrplot")
library(corrplot)
# Corrplot with cor object
corrplot(cor, type = "upper", tl.col = "black", tl.srt = 45)
```

2.2.3 Variant Position

The requirements for a visualisation with KaryoploteR [13] are (1) TXT file with the chromosome structure of the reference genome, (2) VCF file (single or merged), (3) BED file of genomic regions of interest (optional). The VCF file to be analysed is generated with the DArTseq pipeline. Multiple VCF files can also be merged together for combined analysis. Regions of interest in BED format can be generated as explained in the **3 Phylogenetics** section below. A text file with chromosome structure can easily be manually constructed. The VCF file to be analysed contains the needed information. It contains the chromosome ID and length in basepairs. The text file has the following structure:

chr01	1	4356092
chr02	1	3921877
chr03	1	4561284
...	...	...

The RStudio script below explains the installation of the KaryoploteR package and the subsequent analysis. The SNPs can be plotted in various ways: points, density, region-based, coverage.

```
# Install and load packages
install.packages("BiocManager")
BiocManager::install("karyoploteR")
library(karyoploteR)
# Read VCF and construct genome
vars <- read.table("VCF.vcf")
custom.genome <- toGRanges("ChromStructure.txt")
vars <- toGRanges(vars[,c(1,2,2,3:length(vars))])
# Plot variants as points
kp <- plotKaryotype(genome=custom.genome, plot.type=6)
kpPoints(kp, data=vars, y=0.5)
# Plot variants as density
kp <- plotKaryotype(genome=custom.genome, plot.type=1)
kpPlotDensity(kp, data=vars)
# Plot genomic regions
regions.reads <- toGRanges("GENE.bed")
kp <- plotKaryotype(genome=custom.genome)
kpPlotRegions(kp, data=regions.reads)
# Plot gene of interest-related variants
regions.gene <- toGRanges("GENE.bed")
kp <- plotKaryotype(genome=custom.genome, chromosomes="chr*")
kpPlotRegions(kp, data=regions.gene)
# Plot the per base coverage (of gene of interest)
regions.coverage <- toGRanges("GENE.bed")
kp <- plotKaryotype(genome=custom.genome, chromosomes="chr*")
kpPlotCoverage(kp, data=regions.coverage, border="blue", col="orchid")
```

Besides the visualisations above, the construction of a Manhattan plot is also possible, visualising the strongest associations across all the SNPs of a sample or sample set. This also uses the RStudio package `regioneR` [14].

```
# Install and load packages
BiocManager::install("regioneR")
library(regioneR)
library(karyoploteR)
# Load VCF and construct genome
vcf <- read.table("VCF.vcf")
custom.genome <- toGRanges("ChromStructure.txt")
# Extract SNP values
snps <- toGRanges(vcf[,c(1,2,2)])
snps$pval <- rnorm(n=NROW(snps), mean=0.5, sd=1)
snps$pval[snps$pval<0] <- -1*snps$pval[snps$pval<0]
snps$pval <- 10^(-1*snps$pval)
snps.ranges <- toGRanges(snps)
# Construct Manhattan plot
```

```
kp <- plotKaryotype(custom.genome, plot.type=4)
transf.pval <- -log10(snps.ranges$pval)
points.col <- colByValue(transf.pval, colors=c("#BBBBBB00", "grey"))
kp <- kpPlotManhattan(kp, data=snps.ranges, highlight="GENE.bed", \
        highlight.col="red", points.col=points.col)
ymax <- kp$latest.plot$computed.values$ymax #y-axis
ticks <- c(0, seq_len(floor(ymax)))
kpAxis(kp, ymin=0, ymax=ymax, tick.pos = ticks)
```

3 Phylogenetics

Discussed in this section is the reconstruction of evolutionary history of samples based on molecular (sequence) data. First, the VCFs need to be combined into one multi-sample VCF file. Then, the VCF file containing all the SNP data needs to be filtered to remove SNPs with low quality or lacking informative content. Then we discuss the selection of neutral SNPs (inter- and intragenic regions) as opposed to non-neutral SNPs (exons). The use of extra steps discussed in **3.3 The Missing Data Problem** and **3.4 The Fake Marker Problem** combined with high-quality algorithms as found in MrBayes (Bayesian), RAxML-NG (Maximum Likelihood), and IQ-TREE (Maximum Likelihood), is highly advised. The **3.7 "Quick" Phylogenetics** section addresses a fast method to generate a first basic tree. Be careful, this method only shows a rudimentary result.

3.1 SNP Filtering

While bcftools [4] and vcftools [9] take VCFs as direct input, useful in adapting and modifying VCF files, the *libvcflib* packages [15] are often used by third-party applications to parse and manipulate VCFs. The installation of bcftools is shown in the **1.2.2 Dependencies and Tools** section.

Ubuntu terminal

```
# Install packages
sudo apt install vcftools libvcflib-tools libvcflib-dev
# For macOS/OSX
brew install vcftools brewsci/bio/vcflib
```

The bash script below handles all the VCF files that are present in the directory of execution. Be careful no other, unwanted VCF files with the same extension (./*snp.vcf) are present in the directory, or they will be included! The script bgzips the files and generates an index, both necessary for merging into a multi-sample VCF file with bcftools.

The *-0* argument in the *bcftools merge* command assumes the genotypes at missing sites are 0/0 (both alleles the same as reference), and prevents potential 'NA/NaN/Inf' errors in the subsequent analyses. The *-Oz* argument defines a bgzipped output named with the *-o* argument (full name with extension).

Bash script

```
#!/bin/bash
for file in ./*snp.vcf; do
        bgzip $file
done
for file in ./*snp.vcf.gz; do
        tabix -f -h $file
done
bcftools merge ./*snp.vcf.gz --threads 4 -0 -Oz -o VCF.vcf.gz
```

The filtration steps are a combination of bcftools, vcftools, and GATK4 SelectVariants. Removal of the genotype likelihood (GL) tag is required for GATK SelectVariants. The first filter step includes the minimum mean read depth (–min-meanDP), the minor allele count (–mac), the minimum site quality(–minQ), and the minimum genotype quality (–minGQ). The first GATK step excludes the non-variant calls (1/1 or ./. in every sample) and the second GATK step removes the multi-allelic SNPs (more than two alleles are probably read errors). The last vcftools step removes variants based on the Minor Allele Frequency (MAF; –maf) and the proportion of missing data, with 0 meaning sites can be completely missing (–max-missing).

```
# remove genotype likelihood tag to prevent errors in next step
bcftools annotate -x 'FORMAT/GL' VCF.vcf > VCF_GL.vcf
# filter based on minimum mean depth, minimum quality, and minor allele count
vcftools --vcf VCF_GL.vcf --min-meanDP 5 --mac 2 --minQ 20 --recode \
# remove 1/1 or ./. non-variant calls
gatk SelectVariants -R REF.fasta -V VCF_GL_VT.recode.vcf --exclude-non-variants \
                -O VCF_GL_VT_ENV.vcf
# remove multi-allelic variants (most likely read errors)
gatk SelectVariants -R REF.fasta -V VCF_GL_VT_ENV.vcf \
        --restrict-alleles-to BIALLELIC -O VCF_GL_VT2_ENV_RA.vcf
# filter on minor allele frequency and maximum missing data
vcftools --vcf VCF_GL_VT_ENV_RA.vcf --maf 0.05 --max-missing 0.05 --recode \
        --out VCF_filtered
```

Many additional options for SNP filtering are available. Check the references for bcftools [4], GATK [7], and vcftools [9] for more information and adapt the filtering steps according to your needs and preferences.

3.2 Neutral Genetic Markers

Assessments of population genetics and historical demographics have traditionally been based on neutral markers while explicitly excluding adaptive markers. The DArT restriction enzymes see no difference if the restriction sites are located in exons or introns. Therefore, the difference between exonic and intronic SNPs has to be made.

Solution Neutral SNPs can easily be selected when an annotation file of the used reference is available. This annotation file can be converted into genomic coordinates to select (or remove) SNPs that fall within those ranges. Annotation files are often in *gff3* (General Feature Format 3) format. With the BEDOPS utilities [16], this can easily be converted to BED format, which can be used by the *bcftools filter* function to select SNPs within or outside those regions. For BEDOPS, follow the installation on:
`https://bedops.readthedocs.io/en/latest/content/installation.html#linux`
The *-R* flag in the *bcftools filter* command select SNPs within the regions specified in the BED file.

```
# BEDOPS command for GFF3 to BED
gff2bed --input=gff --output=bed < GFF3.gff3 > GFF3.bed
# Index VCF
bgzip VCF_filtered.recode.vcf
tabix -f -h VCF_filtered.recode.vcf.gz
# Filter VCF
bcftools filter -R GFF3.bed -Oz VCF_filtered.recode.vcf.gz > VCF_neutral.vcf.gz
```

3.3 The Missing Data Problem

DArTseq services offer multiple analysis depths (low, medium, and high), resulting in a varying amount of reads generated. Thus the number of positions with generated reads can vary between samples from different service types. This causes wrongful interpretation of missing reads/SNPs in samples analysed with lower depth in comparison to samples analysed with higher depth.

Solution The VCF can be filtered based on reads present in 90-99% of the samples (depending on the allowed percentage of missing data). Within the SMAP toolkit [17], the delineate tool compares all BAM files (in BAMs/ folder) and with, for example, 'completeness' (-w) of 0.95 only selects positions where 95% of samples have at least 3 mapped reads (-x 3) of quality 20 (-q 20). The output is a BED file with which the multi-sample VCF can be filtered. This can also be performed before the steps explained in the **3.1 SNP Filtering** section. The *smap delineate* command requires the BAM files of all samples in one location, here in the *BAMs* directory.

```
# SMAP installation
pip install --upgrade pip
pip install ngs-smap
# SMAP for detecting read positions
smap delineate BAMs/ -r stranded -p 4 -q 20 -s 20 -x 3 -y 1500 -w 100
```

If SMAP encounters the `outofmemory` error, the amount of BAM files it tries to process requires an amount of memory which is too large for the system. The BAM files of the samples can be split into chromosomes with the *bamtools split* function of BamTools [18]. Each chromosome is then analysed separately by *smap delineate*, and the resulting BED files can be merged to represent the entire genome. Run the script below in the directory containing all the BAM files. After bgzipping and indexing the multi-sample VCF, it can be filtered with the merged BED file using the *bcftools filter* function as explained at the end of **3.2 Neutral Genetic Markers**. In the script, the unmapped reads are removed before splitting the BAM files into chromosomes. The seperate chromosome BAM files are gathered in one directory for each chromosome (including chloroplast and mitochondrion) to prevent clogging up the current directory with new files. After generating the *CompletePositions.bed* output, all additional files created with this script can be deleted if not needed anymore.

```
#!/bin/bash
# remove unmapped reads
for file in ./*bam; do
    samtools view -b -F 4 $file > "${file%.bam}.mapped.bam"
done
# split BAMs into reference segments
for file in ./*mapped.bam; do
        bamtools split -in $file -reference
done
# clean up directory
mkdir CHLORO
mkdir MITO
mv *mitochondrion* ./MITO/
mv *chloroplast* ./CHLORO/
for i in {01..11}; do
        mkdir "CHROM$i"
        mv *chr$i* ./"CHROM$i"/
done
# index all the new BAMs
for file in ./*/*bam; do
        samtools index $file
done
# smap delineate for each reference segment
smap delineate MITO/ -r stranded -p 4 -q 20 -s 20 -x 3 -y 1500 -w 95
mv *Set1_C95.0_SMAP20_CL0_inf.bed "ReadPositionsMITO.bed"
smap delineate CHLORO/ -r stranded -p 4 -q 20 -s 20 -x 3 -y 1500 -w 95
mv *Set1_C95.0_SMAP20_CL0_inf.bed "ReadPositionsCHLORO.bed"
for i in {01..11}; do
        smap delineate "CHROM$i"/ -r stranded -p 4 -q 20 -s 20 -x 3 -y 1500 -w 95
        mv *Set1_C95.0_SMAP20_CL0_inf.bed "ReadPositionsChr$i.bed"
done
# merging BEDs into one
bedops --merge ReadPositions* > CompletePositions.bed
```

3.4 The Fake Marker Problem

Phylogenetic tools like IQ-TREE, MrBayes, and RAxML-NG use sequence alignments for analysis. The VCF files that are generated and worked with here are not sequence alignments. Generally, the SNPs are concatenated into one string, i.e. one false/unreal marker, and aligned for each sample. A multi-sample (X) VCF with 5000 SNPs thus becomes a 5000 basepair sequence alignment with X samples. A bad representation of reality at best.

Solution The SMAPapp_Alignment script written by Yves Bawin takes from the BAM files the processed FASTQ read sequences for which there are SNPs in the multi-sample VCF file and aligns those to use as hundreds true/real markers. SMAPapp_Alignment is based on specific versions of Python modules. To prevent downgrading modules necessary for other processes, creating a virtual environment is recommended. First, download the necessary packages and then create the virtual environment based on python3.8. Here the created environment is called pyvenv3.8 (PYthon Virtual ENVironment 3.8). Lastly, activate the environment and install the necessary versions of the modules needed.

```
# Work in the home directory
cd
# Install packages
sudo apt install python3.8 python3.8-dev python3.8-venv
sudo apt install bedtools libcurl4-openssl-dev
# Create python3.8 virtual environment
mkdir venv
```

```
python3.8 -m venv venv/SMAPapp_Alignment
# Activate environment
source venv/SMAPapp_Alignment/bin/activate
# Install modules==version
pip install biopython==1.79
pip install natsort==7.1.1
pip install pysam==0.18.0
pip install pybedtools==0.9.0
# Leave virtual environment
deactivate
```

The SMAPapp_Alignment tool required a BED file to specify the genomic regions for which reads may be aligned. The BED file is generated through *smap delineate* (see **3.3 The Missing Data Problem**). The *–names* argument refers to a text file with 3 columns (tab-delimited): [1] sample name, [2] BAM file location, [3] data type. See an example below.

```
# Example --names file
# SE for single end data
Musa_lutea_VTN05    ./BAMs/Musa_lutea_VTN05.bam     GBS:SE
Musa_lutea_VTN07    ./BAMs/Musa_lutea_VTN07.bam     GBS:SE
Musa_lutea_VTN09    ./BAMs/Musa_lutea_VTN09.bam     GBS:SE
```

```
# Generate BED
smap delineate BAMs/ -r stranded -p 2 -q 20 -x 5 -y 1500 -w 5 -c 10 -s 20
# Activate pyvenv3.8
source pyvenv3.8/bin/activate
# SMAPPapp_Alignment command (default PHYLIP output)
python3 SMAPapp_Alignment.py --bed BED.bed --sample-type GBS:SE --vcf VCF.vcf \
        --names names.txt --reference REF.fasta --ploidy 2 --out VCF_Alignment \
        --sequence_type all_sites --delete_intermediate_files
# SMAPPapp_Alignment command (NEXUS output)
python3 SMAPapp_Alignment.py --bed BED.bed --sample-type GBS:SE --vcf VCF.vcf \
        --names names.txt --reference REF.fasta --ploidy 2 --out VCF_Alignment \
        --sequence_type all_sites --format Nexus --delete_intermediate_files
```

The default output are multiple PHYLIP alignment files which can used directly as input for phylogenetic tools like RAxML-NG and IQTree. MrBayes (and also IQTree) takes NEXUS alignments as input. Use the `--formatNexus` argument to change the output format. The next step includes running a phylogenetic analysis for each generated PHYLIP/NEXUS alignment with a tool of choice.

3.5 Phylogenetic Tools

Discussed here are the basic commands and analyses for MrBayes (Bayesian), RAxML-NG (Maximum Likelihood), and IQ-TREE (Maximum Likelihood). Additional interesting parameters are discussed, as well as simple bash script for automation of phylogenetic analyses across multiple alignments.

3.5.1 Bayesian Inference of Phylogeny

Bayesian inference is a method of statistical inference used to update the probability for a hypothesis as more evidence or information becomes available, particularly important in the dynamic analysis of a sequence of data. Bayesian inference of phylogeny combines the information in the prior and in the data likelihood to create the so-called posterior probability of trees, which is the probability that the tree is correct given the data, the prior and the likelihood model. A prior represents your prior beliefs about certain parameter before observation of the data. MrBayes is a program used for Bayesian inference of phylogeny and model choice across a wide range of phylogenetic and evolutionary models [19]. MrBayes optimizes tree topology, branch lengths, and model parameters using Metropolis-Coupled Markov-Chain Monte-Carlo approach or $(MC)^3$. The commands below install MrBayes in `/home/USER/MrBayes/src/`. Add this location to the `$PATH` in the `.bashrc` file.

```
# Download and install MrBayes
git clone --depth=1 https://github.com/NBISweden/MrBayes.git
cd MrBayes
./configure
make && sudo make install
# Add to .bashrc file from home directory
nano $HOME/.bashrc
```

Starting MrBayes will prompt the MrBayes > command line. First, load the data set in NEXUS format into MrBayes with the `execute` command (or simply `exe` for short). Note that the input file must be located in the same folder (directory) where MrBayes was run. Otherwise, specify the path to the input file.

```
# Start MrBayes
mb
# Load data set
> execute VCF.nexus
```

Two commands, `lset` and `prset`, specify the evolutionary model to use during the analysis. The former defines the structure of the model, while the latter defines the prior probability distributions on the parameters of the model. In this example, the model is $GTR + I + \Gamma$(General Time Reversible model with a proportion of invariable sites and a gamma-shaped distribution of rates across sites). To check the model before running the analysis, use the `showmodel` command. It provides an overview of the settings.

```
# Commands info
> help lset
> help prset
# Set GTR model
> lset nst=6
# Set gamma-shaped distribution
> lset rates=invgamma
```

```
# Check model
> showmodel
```

The analysis is started by issuing the `mcmc` command. The `helpmcmc` command gives an overview of the default run settings. As a test, decrease the number of generations, increase the diagnostic frequency, decrease how often the chain is sampled, and increase the frequency brief info about the analysis is printed to the screen.

```
# Command info
> help mcmc
# Run analysis
> mcmc ngen=20000 diagnfreq=1000 samplefreq=100 printfreq=100
```

At the end of the run, MrBayes prompts whether or not to continue with the analysis. The main determining factor is the average standard deviation of split frequencies. By default, MrBayes runs two simultaneous, completely independent analyses starting from different random trees (`mcmc` argument `nruns=2`). As the two runs converge onto the stationary distribution, the average standard deviation of split frequencies approaches zero, reflecting the fact that the two random tree samples become increasingly similar. A value below 0.01 is very good, but values between 0.01 and 0.05 may already be adequate depending on the purpose of the analysis.

Another determining factor is the Effective Sample Size (ESS). The ESS of a parameter sampled from an MCMC (such as BEAST or MrBayes) is the number of effectively independent draws from the posterior distribution that the Markov chain is equivalent to. During the run, samples of the substitution model parameters have been written to the .p files every samplefreq generation. The simplest way to calculate the ESS is to load the .p log files into Tracer (https://github.com/beast-dev/tracer)(**Figure 4**) [20]. The larger the better, if the ESS of a parameter is small then the estimate of the posterior distribution of that parameter will be poor. Generally, an ESS larger than 200 is a good indication.

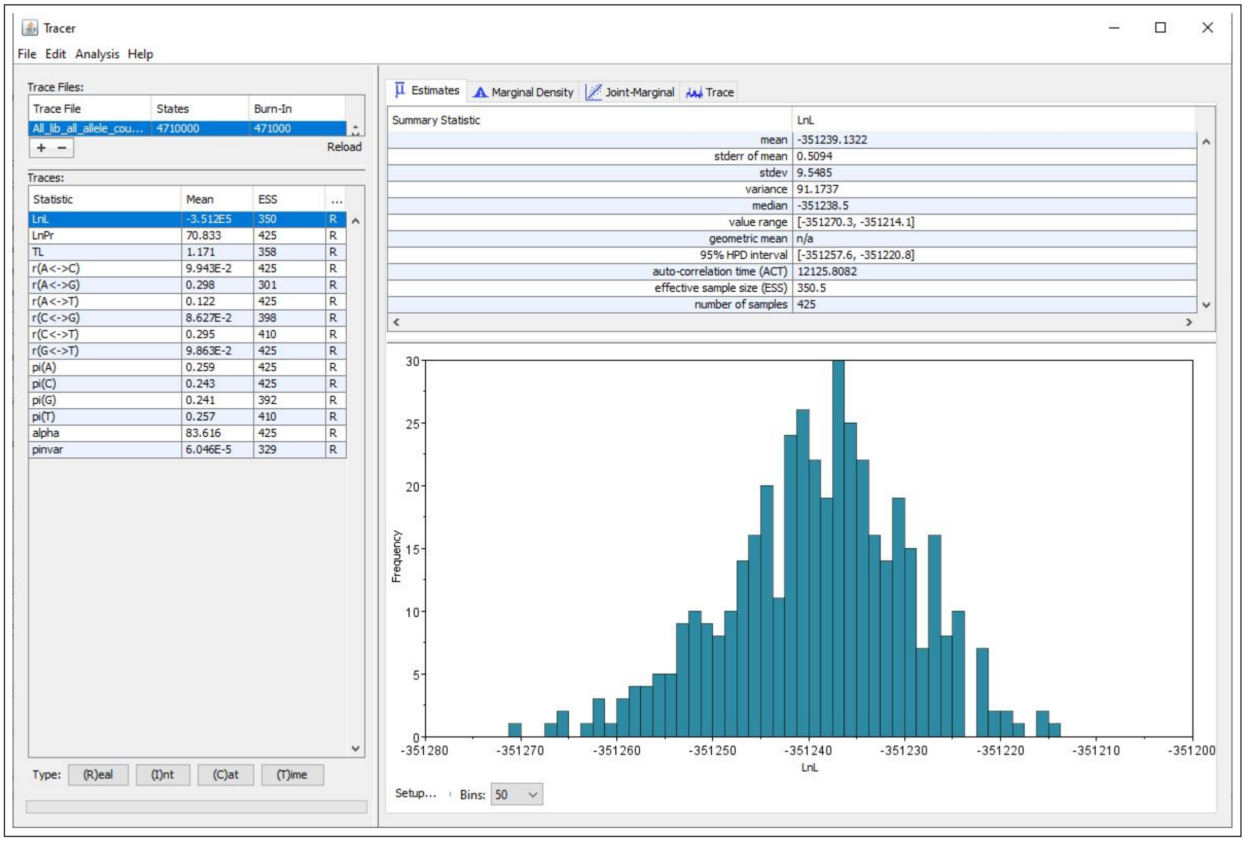
The tab-delimited .p files can also be summarised with the `sump` command in the MrBayes prompt. By default, it uses the same burn-in as the convergence diagnostics in the `mcmc` command.

```
# Summarise sampled parameters values
> sump
```

Trees and branch lengths are printed to the nexus-formatted run1.t and run2.t files. These can be summarised with the `sumt` command. It will output summary statistics for the taxon bipartitions, a tree with clade credibility (posterior probability) values, and a phylogram (if branch lengths have been saved). The clade tree gives the probability of each partition or clade in the tree. The phylogram gives the branch lengths measured in expected substitutions per site. Also, additional summary and statistics files are generated. The .con.tre file includes the consensus tree in a suitable format for visualisation with FigTree (or other software).

```
# Summarise tree samples
> sumt
```

Automation of MrBayes analyses requires the use of a `batch` file. Similar to setting parameters through `lset`, `prset`, and `mcmc`, the `batch` file contains all the analysis information: input file, priors, model, run info, etc. Also below is a simple bash script that searches the directory for all NEXUS files. It then creates a temporary file named



Trace File	States	Burn-In
All_lib_all_allele_cou…	4710000	471000

Statistic	Mean	ESS	…
LnL	-3.512E5	350	R
LnPr	70.833	425	R
TL	1.171	358	R
r(A<->C)	9.943E-2	425	R
r(A<->G)	0.298	301	R
r(A<->T)	0.122	425	R
r(C<->G)	8.627E-2	398	R
r(C<->T)	0.295	410	R
r(G<->T)	9.863E-2	425	R
pi(A)	0.259	425	R
pi(C)	0.243	425	R
pi(G)	0.241	392	R
pi(T)	0.257	410	R
alpha	83.616	425	R
pinvar	6.046E-5	329	R

Summary Statistic	LnL
mean	-351239.1322
stderr of mean	0.5094
stdev	9.5485
variance	91.1737
median	-351238.5
value range	[-351270.3, -351214.1]
geometric mean	n/a
95% HPD interval	[-351257.6, -351220.8]
auto-correlation time (ACT)	12125.8082
effective sample size (ESS)	350.5
number of samples	425

Bins: 50

Figure 4: Example of the Tracer interface [20] when analysing the MrBayes run1.p and run2.p log files.

`temp_file.nex` which the MrBayes `batch` file (also in NEXUS format) recognizes as input. In the loop after the MrBayes analysis, every new file with the temp_file prefix placeholder is renamed to contain the original NEXUS file prefix. At the end, any remaining temp_files are removed.

```
#!/bin/bash
for file in ./*nexus; do
    cp $file temp_file.nex
        mb MrBayesBatch.nex
        mv temp_file.run1.p "${file%.nexus}.run1.p"
        mv temp_file.run2.p "${file%.nexus}.run2.p"
        mv temp_file.pstat "${file%.nexus}.pstat"
        mv temp_file.run1.t "${file%.nexus}.run1.t"
        mv temp_file.run2.t "${file%.nexus}.run2.t"
        mv temp_file.con.tre "${file%.nexus}.con.tre"
        mv temp_file.trprobs "${file%.nexus}.trprobs"
        mv temp_file.lstat "${file%.nexus}.lstat"
        mv temp_file.tstat "${file%.nexus}.tstat"
        mv temp_file.vstat "${file%.nexus}.vstat"
        mv temp_file.txt "${file%.nexus}.txt"
        mv temp_file.ckp "${file%.nexus}.ckp"
        mv temp_file.mcmc "${file%.nexus}.mcmc"
        mv temp_file.parts "${file%.nexus}.parts"
done
rm -r temp_file*
```

```
#NEXUS
BEGIN mrbayes;
[**************************************]
[Read in the data and set preliminaries]
[**************************************]
log start filename=temp_file.txt;           [log output to a text file]
set autoclose=yes nowarn=yes;               [needed to run in batch mode]
execute temp_file.nex;                      [read in the data file]
[*****************************]
[Analysis with MCMC parameters]
[*****************************]
lset nst=6 rates=invgamma;

mcmc data=yes nruns=2 nchains=4         [specify sampling with data]
ngen=20000 printfreq=100 samplefreq=100 [details of the MCMC analysis...]
diagnfreq=1000 diagnstat=maxstddev      [...and the diagnostics of MCMC]
filename=temp_file;           [output filename]
sump filename=temp_file;      [summarize model parameters]
sumt filename=tempfile;      [summarize samples of trees]
quit;
END;
```

Interesting Parameters By default, the **burnin** fraction (burninfrac) is set to 25%, discarding this percentage of samples from the beginning of the chain every time the diagnostics are calculated. The relburnin setting determines whether a fixed burnin (relburnin=no) or a burnin percentage (relburnin=yes) is used. Depending on the data and analysis, 10% burnin may already suffice as 25% removes a fraction of high quality trees in the 10% to 25% interval. Therefore, checking log files in Tracer is an important indicator towards adapting the burnin fraction to an optimal value.

The ngen setting is the **number of generations** for which the analysis will be run. It is useful to run a small number of generations first to make sure the analysis is correctly set up and to get an idea of how long it will take to complete a longer analysis. The default setting is 1.000.000 generations, but is often too little a number. Around 10.000.000 generations is a better start, keeping an eye on the average standard deviation of split frequencies, and checking the run1.p and run2.p log files in Tracer. As a rule of thumb, some people use 1.000.000 generations per marker in the analysis. For SNPs this is different, as they are concatenated and represented as one large marker.

The `nucmodel` option specifies the general form of the **nucleotide substitution model**. The options are "4by4" (the standard model of DNA substitution in which there are only four states (A,C,G,T/U)), "doublet" (a model appropriate for modelling the stem regions of ribosomal genes where the state space is the 16 doublets of nucleotides), "codon" (the substitution model is expanded around triplets of nucleotides–a codon), and "Protein" (triplets of nucleotides are translated to amino acids, which form the basis of the substitution model).

`Nst` sets the number of **substitution types**: "1" constrains all of the rates to be the same (e.g., a JC69 or F81 model); "2" allows transitions and transversions to have potentially different rates (e.g., a K80 or HKY85 model); "6" allows all rates to be different, subject to the constraint of time-reversibility (e.g., a GTR model). Finally, `nst` can be set to 'mixed', which results in the Markov chain sampling over the space of all possible reversible substitution models, including the GTR model and all models that can be derived from it by grouping the six rates in various combinations. This includes all the named models above and a large number of others, with or without name.

The `treeagepr` parameter specifies the **prior probability distribution** on the tree age when a uniform or fossilization prior is used on the branch lengths of a clock tree. The options are: `prsettreeagepr=fixed(), uniform(,),offsetexponential(,),truncatednormal(,,),lognormal(,),offsetlognormal(,,),gamma(,),offsetgamma(`
`)`. These are the same options used for the 'Calibrate' command. Note that, unlike elsewhere in MrMayes, we always use the mean and standard deviation of the resulting age distribution rather than the standard parameterization, if different. This is to facilitate for the users who want to focus on the information conveyed about the age. For those who wish to use the standard parameterization, there are simple conversions between the two. See the 'Calibrate' command for more information. The tree age is simply the age of the most recent common ancestor of the tree. If the clock rate is fixed to 1.0, which is the default, the tree age is equivalent to the expected number of substitutions from the root to the tip of the tree, that is, tree height. The tree age prior ensures that the joint probability for the uniform prior (or fossilisation prior) model of branch lengths on a clock tree is proper. The default setting is `gamma(1,1)`. If the root node in the tree is calibrated, the root calibration replaces the tree age prior.

The `clockratepr` option specifies the prior assumptions concerning the **base substitution rate** of the tree, measured in expected number of substitutions per site per time unit. The default setting is 'Fixed(1.0)', which effectively means that the time unit is the number of expected substitutions per site. If you do not have any age calibrations in the tree, you can still calibrate the tree using `clockratepr`. For instance, if you know that your sequence data evolve at a rate of 0.20 substitutions per million years, you might calibrate the tree by fixing the substitution rate to 0.20 using `prsetclockratepr=fixed(0.20)` after which the tree will be calibrated using millions of years as the unit. You can also assign a prior probability distribution to the substitution rate, accommodating the uncertainty of it. When you calibrate the nodes, you should properly set this prior to match the time unit of the calibrations. You can choose among normal, lognormal, exponential and gamma distributions for this purpose. For instance, to assign a normal distribution truncated at 0, so that only positive values are allowed, and with mean 0.20 and standard deviation of 0.02, you would use `prsetclockratepr=normal(0.20,0.02)`. The lognormal distribution is parameterized in terms of the mean and standard deviation on the log scale (natural logs). For instance, `prsetclockratepr=lognormal(-1.61,0.10)` specifies a lognormal distribution with a mean of log values of -1.61 and a standard deviation of log values of 0.10. In such a case, the mean value of the lognormal distribution is equal to $e^{(-1.61+0.10^2/2)} = 0.20$ Note that the `clockratepr` parameter has no effect on non-clock trees.

3.5.2 Randomised Axelerated Maximum Likelihood

In statistics, maximum likelihood estimates the parameters of an assumed probability distribution, maximizing a likelihood function so that, under the assumed statistical model, the observed data is most probable. RAxML (Randomized Axelerated Maximum Likelihood) is a popular program for phylogenetic analysis of large datasets under maximum likelihood. Its major strength is a fast maximum likelihood tree search algorithm that returns trees with good likelihood scores. RAxML Next Generation (RAxML-NG) [21] offers improvements in speed, flexibility and user-friendliness. RAxML-NG optimizes tree topology using a variant of iteratively subtree pruning and regrafting (SPR) called lazy subtree rearrangement (LSR). It also optimizes branch lengths using the Newton-Raphson method and optimizes model parameters using Brents algorithm.

From the GitHub page (https://github.com/amkozlov/raxml-ng), a pre-built binary can be downloaded. Extract the zipped archive, and the resulting folder (-d RAxML-NG) contains the `raxml-ng` executable for Unix/Linux systems. Add this location to the $PATH for convenience. RAxML-NG is installed in $HOME/USER/RAxML-NG/ when following the instructions below. Replace the '1.X' with the current version of RAxML-NG.

```
# Install dependencies
sudo apt-get install cmake flex bison libgmp3-dev
# Move archive into home directory
cd
mv /mnt/c/Users/USER/Downloads/raxml-ng_v1.X_linux_x86_64.zip .
# Extract archive
unzip raxml-ng_v1.X_linux_x86_64.zip -d RAxML-NG/
# Open .bashrc to add to $PATH
nano $HOME/.bashrc
```

The input alignment format of RAxML is relaxed interleaved, sequential PHYLIP, or FASTA. Relaxed means that sequence names can be of variable length between 1 up to 256 characters. The optional input tree format is NEWICK, and must not always be comprehensive, i.e., need not contain all taxa of the alignment.

```
# RAxML command
raxml-ng --msa VCF.phy --data-type DNA --prefix PREFIX --log DEBUG --model GTR+G
```

Another standard task is to evaluate trees, i.e., compute the likelihood of a given fixed tree topology by just optimizing model and/or branch length parameters on that fixed tree. This is frequently needed in model and hypothesis testing. The basic option is `--evaluate`. With `--opt-modelon/off` you can enable/disable model parameter optimization. With `--opt-brancheson/off` you can enable/disable branch length optimization.

```
# RAxML Tree Likelihood Evaluation
# First, evaluate under the most simple model
raxml-ng --evaluate --msa VCF.phy --model JC --tree VCF.raxml.bestTree \
            --prefix Eval1
# Add rate heterogeneity
raxml-ng --evaluate --msa VCF.phy --model JC+G --tree VCF.raxml.bestTree \
            --prefix Eval2
```

```
# For the GTR model
raxml-ng --evaluate --msa VCF.phy --model GTR --tree VCF.raxml.bestTree \
            --prefix Eval3
raxml-ng --evaluate --msa VCF.phy --model GTR+G --tree VCF.raxml.bestTree \
            --prefix Eval4
```

Finally, you can re-use the optimized model parameters obtained in a previous RAxML-NG run:

```
# RAxML command with model from previous run
raxml-ng --msa VCF.phy --model OldRun.raxml.bestModel --prefix NewRun --log DEBUG
```

If you know what you're doing, you can disable all or some of the RAxML-NG builtin safeguards with the `--force` option. This option also accepts a comma-separated list of individual checks, e.g. `--forcemsa\_dups, msa\_allgaps`.

```
--force msa_names   Do not check for invalid characters in sequence names.
--force msa_dups    Do not check for duplicate sequences.
--force msa_allgaps Do not check for sequences and columns consisting entirely of
                    missing data (gaps).
--force msa         Disable all MSA-related checks listed above.
...
```

For automation, check the loop in the bash script below.

```
#!/bin/bash
for file in ./*phy; do
    raxml-ng --msa $file --data-type DNA --prefix "${file%.phy}" --model GTR+G \
            --log DEBUG
done
```

3.5.3 IQTREE

The IQTree software [22] is a time and search efficient maximum likelihood tree reconstruction program. It shows a better performance with respect to the maximum likelihood search than RAxML or PhyML.

```
# Install IQTREE
sudo apt-get install iqtree
```

The main arguments of an IQTree command are listed below, followed by a simple bash script that loops over all PHYLIP alignments in a directory and creates for each one a maximum likelihood tree.

```
# IQTREE arguments
-s | Input alignment in PHYLIP/FASTA/NEXUS/CLUSTAL/MSF format
-st | Datatype: BIN, DNA, AA, NT2AA, CODON, MORPH (default: auto-detect)
-o | Outgroup taxon name for writing .treefile
-pre | Prefix for all output files (default: aln/partition)
-nt | Number of cores/threads or AUTO for automatic detection
-ntmax | Max number of threads by -nt AUTO (default: #CPU cores)
-v | Verbose mode, printing more messages to screen
-keep-ident | Keep identical sequences (default: remove & finally add)
```

```
#!/bin/bash
for file in ./*phy; do
    iqtree -s $file -st DNA -pre "${file%.phy}" -v
done
```

3.6 ASTRAL-III

For multiple gene trees, ASTRAL-III can be used [23]. As stated on the GitHub page: "ASTRAL estimates an unrooted species tree given a set of unrooted gene trees, and is statistically consistent under the multi-species coalescent model (and thus is useful for handling incomplete lineage sorting; ILS). ASTRAL finds the species tree that has the maximum number of shared induced quartet trees with the set of gene trees, subject to the constraint that the set of bipartitions in the species tree comes from a predefined set of bipartitions." For more practical considerations, check Mirarab (2019) [24]. On how ASTRAL can handle multi-allele datasets, check Rabiee *et al.* (2019) [25].

```
# Installation from home directory
cd
sudo apt install openjdk-11-jdk-headless
git clone https://github.com/smirarab/ASTRAL.git
cd ASTRAL/
bash make.sh
```

The hundreds or thousands trees generated with IQTree, MrBayes, RAxML-NG, etc. need to be combined into one file. That file will function as the input for ASTRAL. The `-t2` flag refers to full annotation.

```
# Example concatenate all trees into one file
cat *.tree > AllTrees.trees
# Basic ASTRAL command
java -jar astral.jar -i AllTrees.trees -t 2 -o AllTrees.ASTRAL.tre
```

If ASTRAL encounters a `java.lang.OutOfMemoryError:Javaheapspace` error, the system CPU/RAM is insufficient to process that large amount of trees. One solution is to specify the maximum amount of RAM is available for ASTRAL to use: `java-Xmx12G-jarastral.jar...` Here 12GB of RAM is allocated to ASTRAL. The trees file can also be uploaded to `usegalaxy.be` or `usegalaxy.org` and processed with the ASTRAL-III package.

3.7 "Quick" Phylogenetics

Widely used phylogenetic tools (e.g. MrBayes and RAxML) have long processing times. First, we discuss some ways to generate a cladogram fast to give a quick impression of the results. **Be aware** this only generates a rudimentary result! Check **3.3 The Missing Data Problem** and **3.4 The Fake Marker Problem** for more information why this "quick" phylogenetics is unsuitable for DArTseq data.

3.7.1 NEWICK Tree Format

This file is a mathematical representation of graph-theoretical trees with edge lengths using parentheses and commas. Here discussed is the generation of NEWICK files with the `VCF-kit` tool and a R script using the `SNPRelate` package. Both take the multi-sample VCF file as direct input.

VCF-kit The Variant Call Format Kit offers a collection of tools to perform a variety of operations on the data stored in VCF files [26]. Here discussed is how it turns a multi-sample VCF file into a NEWICK tree. The complete manual, and more information about VCF-kit can be found on the readthedocs website at: https://vcf-kit.readthedocs.io/en/latest/. VCF-kit uses MUSCLE alignments to produce a NJ (neighbour-joining) of UPGMA (unweighted pair group method with arithmetic-mean) tree, with an output in NEWICK format.

```
# Install dependencies and VCF-kit
pip install numpy
pip install VCF-kit
# Install homebrew for linux (and add to $PATH)
/bin/bash -c "$(curl -fsSL \
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
# Install other dependencies
vk setup
# Phylogenetic tree generation
vk phylo tree nj VCF.vcf > VCF.nj.newick
vk phylo tree upgma VCF.vcf > VCF.upgma.newick
```

SNPRelate The R package `SNPRelate` provides a binary format for SNP data utilizing CoreArray Genomic Data Structure (GDS) data files [27]. Together with packages `ape`, `gdsfmt`, `Rcpp`, and `tidyverse`, a tree is generated and written to a NEWICK format file.

RStudio script

```
# Package installation
install.packages("BiocManager")      # only once
BiocManager::install()               # update BiocManager
BiocManager::install(c("gdsfmt"))    # only once
BiocManager::install(c("SNPRelate"))# only once
```

```
install.packages("Rcpp")            # only once
install.packages("ape")             # only once
# Load libraries
rm(list=ls()) # remove the objects in memory
library(Rcpp)
library(gdsfmt)
library(SNPRelate)
library(tidyverse)
# Load the work directory and generate GDS
setwd("~/Users/Documents/Phylogenetics/") # example
snpgdsVCF2GDS("VCF.vcf", "VCF.gds", ignore.chr.prefix="Bchr")
genofile <- snpgdsOpen("VCF.gds")
snpgdsSummary("VCF.gds")
set.seed(1000) #Following https://benbowlab.github.io/Phylogeny.html
# Generate IBS, matrix, and tree
ibs.hc <- snpgdsHCluster(snpgdsIBS(genofile,num.thread=2, autosome.only=FALSE))
dissMatrix  =  snpgdsIBS(genofile , sample.id=NULL, autosome.only=FALSE, \
    remove.monosnp=TRUE,  maf=NaN, missing.rate=NaN, num.thread=2, verbose=TRUE)
snpHCluster =  snpgdsHCluster(dissMatrix, sample.id=NULL, need.mat=TRUE, hang=0.01)
cutTree = snpgdsCutTree(snpHCluster, z.threshold=15, outlier.n=5, n.perm = 5000, \
    samp.group=NULL, col.outlier="red", col.list=NULL, pch.outlier=4, \
    pch.list=NULL, label.H=FALSE, label.Z=TRUE, verbose=TRUE)
rv <- snpgdsCutTree(ibs.hc)
plot(rv$dendrogram,main="Tree")
# Export tree as NEWICK
library(ape)
my_tree<-as.phylo(ibs.hc$hclust)
write.tree(phy=my_tree, file="VCF.newick")
snpgdsClose(genofile) # close the file
```

3.8 Tree Visualisation

The resulting NEWICK trees from phylogenetic analyses, ASTRAL inference, or other methods can be visualised in multiple ways but here two are presented: `FigTree` and an R script using the `GGMap` package.

FigTree Download and install FigTree from its GitHub repository: https://github.com/rambaut/figtree [28]. Unpack the zipped archive and, in the resulting folder, run the FigTree program. Many options for customization are available. For example, annotate nodes and branches with probability (in percentage) by checking the `Nodes` and `Branches` boxes, expanding the drop down menu, and choosing `percentage` as option. Also a version for command line is available. Other tree visualisation softwares are: Interactive Tree of Life (iTol; https://itol.embl.de/), Molecular Evolutionary Genetics Analysis (MEGA; https://www.megasoftware.net/).

GGMap As mentioned in the VCF-kit manual [26, 29], follow the RStudio script below to plot the generated NEWICK tree.

```
# Install packages
install.packages("tidyverse")
source('http://bioconductor.org/biocLite.R')
biocLite(c('ape','phyloseq','ggmap'), suppressUpdates=TRUE)
# Load packages into library
library(tidyverse)
library(ape)
library(ggmap)
library(phyloseq)
# Load newick tree file
tree <- ape::read.tree(paste0("VCF.newick"))
#optionally set an outgroup.
tree <- root(tree,outgroup = "outgroup", resolve.root = T)
# Generate tree
treeSegs <- phyloseq::tree_layout(phyloseq::phy_tree(tree), ladderize = T)
treeSegs$edgeDT <- treeSegs$edgeDT %>% \
    dplyr::mutate(edge.length = ifelse(edge.length < 0, 0, edge.length), \
    xright = xleft + edge.length)
edgeMap = aes(x = xleft, xend = xright, y = y, yend = y)
vertMap = aes(x = x, xend = x, y = vmin, yend = vmax)
labelMap <- aes(x = xright+0.0001, y = y, label = OTU)
# Plot with ggplot
ggplot(data = treeSegs$edgeDT) + geom_segment(edgeMap) +
  geom_segment(vertMap, data = treeSegs$vertDT) +
  geom_text(labelMap, data = dplyr::filter(treeSegs$edgeDT, !is.na(OTU)), \
  na.rm = TRUE, hjust = -0.05) +
  ggmap::theme_nothing() +
  scale_x_continuous(limits = c(min(treeSegs$edgeDT$xleft)-0.15, \
  max(treeSegs$edgeDT$xright)+0.15), expand = c(0,0))
```

4 Metagenomics

The study of the structure and function of entire nucleotide sequences isolated and analyzed from all the organisms (typically micro-organisms) in a bulk sample. When extracting DNA from a silica-dried leaf sample, not only the DNA of the plant is extracted. Also the micro-organisms present on and in the leaf are submitted to DNA extraction. DArTseq also picks up these nucleotide sequences, the restriction enzymes targeting their recognition sites regardless of the sequence origin. After processing the raw FASTQ reads (removing adapters et cetera), the cleaned-up reads can be blasted against a database of micro-organisms to identify what was present on and in the plant leaf sample.

4.1 Classification

The biological classification is here not based on shared macroscopic characteristics, but microscopic similarities in nucleotide sequences. The DArTseq process has metagenomic sequencing as a happy by-product. For the taxonomic classification of microbial species based on small DNA sequences, a wide variey of tools and software is available online. Here, we discuss Kraken2 and Centrifuge, two widely used systems with different classification algorithms. Then, Recentrifuge offers an additional control by checking the confidence score of the classification before visualising results.

4.1.1 Kraken2

Kraken2 [30, 31] examines the k-mers within a query sequence and uses the information within those k-mers to query a database. That database maps k-mers to the lowest common ancestor (LCA) of all genomes known to contain a given k-mer. The GitHub repository can be cloned into your home directory, before running the installation script. Here, the installation directory is specified as the same directory with the cloned GitHub repository. After installation, add the Kraken2 directory to your $PATH in the .bashrc file (then the Kraken2 commands can simply be performed without specifying the installation directory).

Ubuntu terminal

```
# download and installation
cd $HOME/bin
git clone https://github.com/DerrickWood/kraken2
cd kraken2/
./install_kraken2.sh $HOME/bin/kraken2
# open .bashrc to add to $PATH
nano $HOME/.bashrc
```

Custom Database Besides the standard databases, a custom database can easily be built with the kraken2-build `-add-to-library` flag. More information on standard and custom databases can be found in detail on the github page. Below are the main commands for a custom database with fasta files downloaded from, for example, NCBI. The command for adding fasta files to a library can easily be automated with a simple for-loop script.

```
# database installation
kraken2-build --download-taxonomy --db path/to/custom_db
kraken2-build --add-to-library path/to/custom_fasta --db path/to/custom_db
```

```
# build database from all added fasta files in database folder
kraken2-build --build --db path/to/custom_db
```

Bash script

```
#!/bin/bash
for file in ./library/*/*; do
        kraken2-build --add-to-library --db ./custom_db
done
```

The first step in building the database is constructing a seqid2taxid.map file. Before proceeding to the next step, the process can always be cancelled here. For additional info on including or excluding entries in the database, check Issue56 on the Kraken2 GitHub page (https://github.com/DerrickWood/kraken2/issues/56). The seqid2taxid.map file can also be made manually in tab-delimited format:

```
kraken:taxid|[taxid]||[seqid]        [taxid]
kraken:taxid|1071383|NC_035922.1    1071383
kraken:taxid|284811|NC_005782.2     284811
```

Kraken2 Analysis The standard Kraken2 command to classify a set of sequences against a database, uses the kraken2 command as shown below. Many additional parameters are available to finetune the classification process to your preferences or necessities.

```
#kraken2 command for sequence classification
kraken2 --db path/to/custom_db --gzip-compressed --output [prefix] fastq.gz
```

The *–gzip-compressed* flag specifies the .gz input format, and can be either fasta or fastq. Classified or unclassified reads can be written as an output with the *–classified-out*/*–unclassified-out* flag, which uses the output prefix.

Accuracy and precision of the classification process can be finetuned with many parameters. The *–kmer-length* flag can adjust the k-mer length when building the database with the kraken2-build command. When classifying with the kraken2 command, *–minimum-hit-groups* can be increased (default 2) to minimise false positives, and –confidence can be increased to raise the kmer per minimizer requirement for classification (default 0) (**Figure 5**).

Below is the bash script for Kraken2 to handle multiple sequence files in a directory. Notice the adjusted values for *–minimum-hit-groups* and *–confidence*. Change them to suit your analysis. The output is automatically stored in a different folder 'Output' to reduce the clutter in the directory of the sequence files.

```
#!/bin/bash
for file in ./*GBprocesS.fastq.gz; do
    kraken2 --db ./custom_db --minimum-hit-groups 4
        --confidence 0.25 --gzip-compressed
        --output ./Output/"${file%.GBprocesS.fastq.gz}" $file
done
```

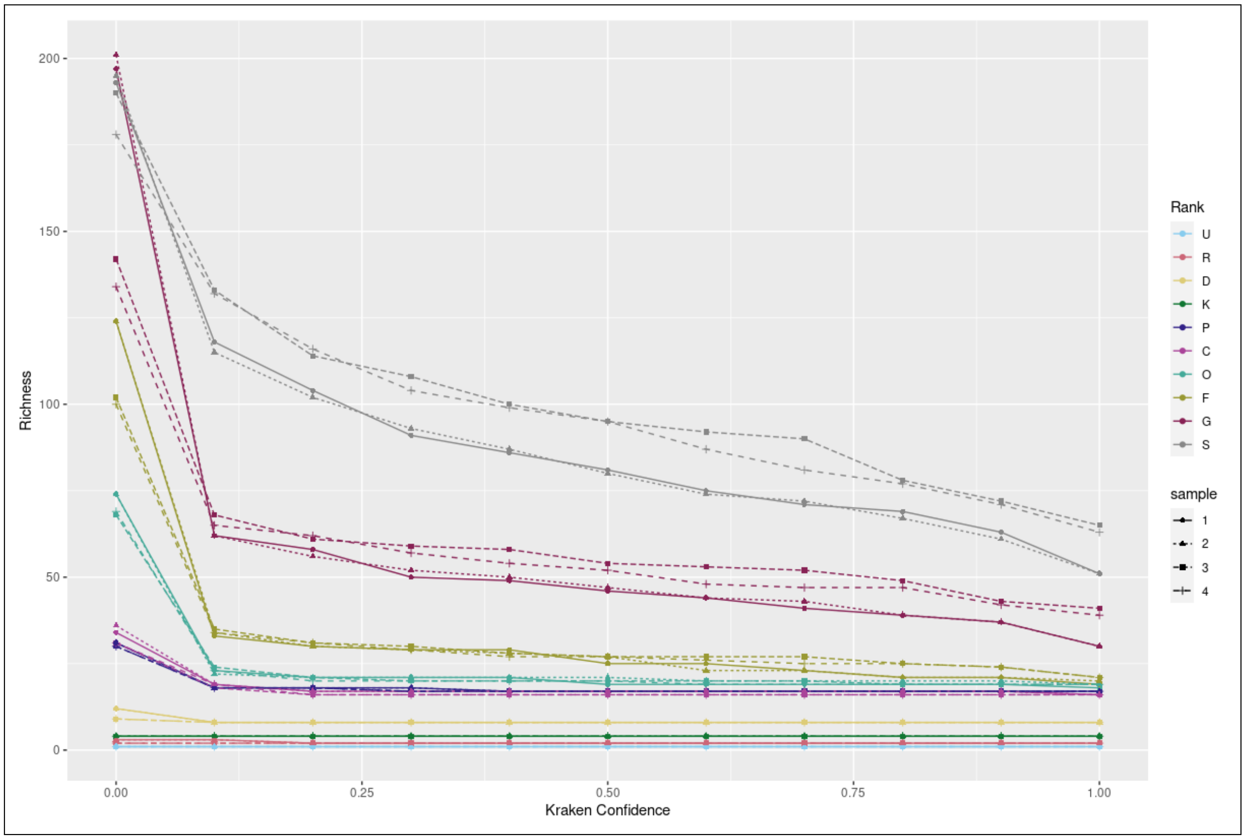


Figure 5: The unique number of taxa at a specific rank (here called richness) assigned by Kraken2 is shown as a function of the confidence parameter. Notice the large drop in richness between confidence 0.0 and 0.1. The –confidence option depends on the context: generally, go with 0.05 or 0.10 for general purposes, or in the case where classifications with $<$ 10 reads really affect the interpretation, use a higher threshold of 0.25 to 0.50.

KrakenTools KrakenTools [32] provides a suite of scripts to be used for post-analysis of Kraken/KrakenUniq/**Kraken2**/Bracken results. For more information about these utilities, refer to the GitHub page: https://github.com/jenniferlu717/KrakenTools/.

Comments

False Results The database entries influence the results heavily. One must find a balance between two states: (1) too few entries increase the amount of false positives, reads get forced into hits when not enough reference material is present, and (2) too much entries divide reads into lower taxonomic ranks that are irrelevant/unlikely to be present (e.g. forma specialis and pathovars). In line with the second statement, Ordonez Roman, N. describes the the forma specialis classification system inadequate and obsolete (DOI: 10.18174/453455). However, to ensure the certainty of the hit, closely related organisms on species level must be included, as to disprove the first statement. Depending on the availability of reference sequences, some organisms will be over- or underrepresented in the database. Example: the complete reference of *Fusarium oxysporum* is overrepresented in comparison with two ITS sequences of *Nigrospora chinensis.*

Disk Space Kraken2 has the downside that the standard databases require a large amount of disk space. Depending on the amount of fasta files in your custom database, this can also quickly take in a lot of disk space.

RSync Error For problems with RSync, or FTP path, check the following GitHub issue where it was resolved with modifying the perl script in the Kraken2 installation: `https://github.com/DerrickWood/kraken2/issues/508`.

4.1.2 Centrifuge

The Centrifuge classifier uses a novel indexing scheme based on the Burrows-Wheeler transform (BWT) and the Ferragina-Manzini (FM) index, optimised specifically for the metagenomic classification problem [33]. This tool is installed from the github repository. Database building is largely done the same as for Kraken2. Don't forget adding the installation directory to PATH in .bashrc. The construction of a custom database uses some of the Kraken2 utilities. Since both tools will generate a custom database, the seqid2taxid.map file created by Kraken2 can also be used by Centrifuge (which doesn't make one on its own). For Centrifuge, the input sequences must be concatenated into one file (command below). Only then can the Centrifuge index be built, and requires nothing more than the created seqid2taxid.map file and the names.dmp and nodes.dmp that are also necessary for 'normal' databases.

```
#installation
cd bin/
git clone https://github.com/DaehwanKimLab/centrifuge
cd centrifuge/
sudo make install prefix=$HOME/bin/centrifuge
#download taxonomy
centrifuge-download -o ./taxonomy taxonomy
#create custom library for custom sequences (.fasta or .fna)
mkdir ./library ./library/bacteria ./library/fungi ./library/viral
cat library/*/* > input-sequences.fna
#build database
centrifuge-build --conversion-table seqid2taxid.map
                    --taxonomy-tree taxonomy/nodes.dmp
                    --name-table taxonomy/names.dmp
                    input-sequences.fna [prefix]
```

Example index build time for 10GB input-sequences.fna (12 processors, 32GB RAM)
Avg bucket size 1.569.670.000 (target 2.060.185.454)
V-sorting samples time = 9 minutes
Ranking v-sort output time = 2 minutes
Invoking L-S on ranks time = 2 minutes
Bucket 1 sorting block time = 28 minutes
Bucket 2 sorting block time = 46 minutes
Bucket 3 sorting block time = 47 minutes
Bucket 4 sorting block time = 36 minutes
Bucket 5 sorting block time = 46 minutes
Bucket 6 sorting block time = 49 minutes
Bucket 7 sorting block time = 19 minutes
Total time for call to driver() for forward index: 05h35m20s

Below is the bash script for Centrifuge to handle multiple sequence files in a directory. For additional options, check the manual at http://www.ccb.jhu.edu/software/centrifuge/manual.shtml.

```bash
#!/bin/bash
for file in ./*GBprocesS.fastq.gz; do
    centrifuge -x [prefix] -U $file -S ./"${file%.GBprocesS.fastq.gz}.centrifuge"
                     --report-file ./report/"${file%.GBprocesS.fastq.gz}.report"
done
```

Comments

False Results Same first point as Kraken2 (see above).

Disk Space Although Centrifuge databases take much less disk space, they have the downside that building databases takes much more time (see example above).

Download Error Problems with downloading databases? Check the issues section of the github page. Here are a couple that could be useful with certain errors: “Centrifuge-download error extra operand”: https://github.com/DaehwanKimLab/

4.1.3 Recentrifuge

Recentrifuge enables the analysis of results from e.g. Centrifuge, Kraken,... (taxonomic classifiers) using interactive pie charts, by placing great emphasis on the confidence level (score) of the taxonomic classifications [34]. The arithmetic of scored taxonomic trees of Recentrifuge supports the 48 taxonomic ranks of the NCBI Taxonomy, including several infraspecific levels such as strain or isolate. If there are one or more negative control samples in the study, Recentrifuge will also generate additional control-subtracted interactive plots with the contamination removed from every sample and from the shared taxa specimen. The novel and robust contamination removal algorithm of Recentrifuge detects and selectively removes various types of pollutants, including crossovers.

```
#installation and dependencies
pip install recentrifuge matplotlib openpyxl pandas
#test installation (can take several hours)
retest
```

The command line for Recentrifuge differs between taxonomic classifiers used to generate output. Below are the ones used for Kraken2 and Centrifuge as these tools are explained above.

```
#centrifuge for single files, or entire directory
rcf -f S1.out -f S2.out -f S3.out
rcf -f output_directory/
#kraken2 for single files, or entire directory
rcf -k S1.krk -k S2.krk -k S3.krk
rcf -k output_directory/
```

Sequences/reads that found a match in the Centrifuge or Kraken2 analysis can be extracted with the rextract command of Recentrifuge (https://github.com/khyox/recentrifuge/wiki).

```
#rextract command (-n gives path to names/nodes.dmp in taxdump folder)
rextract -f file.centrifuge -n /path/to/files.dmp -q fastq
```

4.2 Visualisation

An important output of metagenomics is the (estimation of) abundances of taxonomical or functional groups. These assignments have an inherent uncertainty making it necessary to consider both their hierarchical contexts and their prediction confidence. Recentrifuge already showed additional control in checking confidence scores. The current tools for visualizing metagenomic data are plenty, but need not always be flawless.

4.2.1 KronaTools

Krona charts can be created using KronaTools [35], which includes support for several bioinformatics tools and raw data formats. The interactive charts are self-contained and can be viewed with any modern web browser. Unlike Recentrifuge, Kronatools merely visualises the classification data without comparative analysis and contamination removal. Kronatools is installed with the commands below, for which after installation, the taxonomy needs to be updated.

```
# download and install KronaTools
git clone https://github.com/marbl/Krona
cd ./Krona/KronaTools/
sudo ./install.pl
# update taxonomy
sudo ./updateTaxonomy.sh ./taxonomy
```

Use this bash script for automation of multiple sample analysis (depending on use of Kraken2 or Centrifuge). The -q and -t options specify the columns to use in the metagenomics output file to generate the Krona chart. For both Kraken2 and Centrifuge these columns are the same.

```
#!/bin/bash
for file in ./*_centrifuge; do
    ktImportTaxonomy -q 2 -t 3 $file -o "${file%_centrifuge}_cenKRONA.html"
done
#OR
for file in ./*_kraken2; do
    ktImportTaxonomy -q 2 -t 3 $file -o "${file%_kraken2}_kraKRONA.html"
done
```

4.2.2 WebOfLife & metagenomeSeq

The Python script from WebOfLife (WoL) [36] is used to convert Centrifuge output files to .tsv format. This, in turn, is converted into biom format with the Python *biom convert* command. The 'done' folder specifies all the Centrifuge analysis files, and 'out' is the prefix of the output files of the ogu_from_maps.py script. The remaining analysis is done in RStudio with the metagenomeSeq package [37, 38].

```
#installation
sudo apt install python3-biom-format
sudo apt install python3-h5py
git clone https://github.com/biocore/wol
#biom conversion and OTU table
python3 /wol/code/scripts/ogu_from_maps.py -m centrifuge -t seqid2orgid.map /directory/done/
biom convert -i out.all.tsv -o out.all.biom --table-type="OTU table" --to-hdf5
```

```
BiocManager::install(c("metagenomeSeq")) #only necessary once
BiocManager::install(c("biomformat")) #only necessary once
library(metagenomeSeq)
library(biomformat)
biom <- loadBiom("out.all.biom")
p = cumNormStatFast(biom)        #NORMALISATION
biom = cumNorm(biom, p = p)
mat = MRcounts(biom, norm = TRUE, log = TRUE)
exportMat(mat, file = "tmp.tsv") #EXPORT DATA
count_data <- metagenomeSeq:::extractMR(biom)        #COUNT DATA
count_data <- count_data$counts
obj <- newMRexperiment(count_data) #NEW MR EXP CLASS OBJECT
heatmapColColors=brewer.pal(12,"Set3");
heatmapCols = colorRampPalette(brewer.pal(9, "RdBu"))(10)
plotMRheatmap(obj = obj, #MR CLASS OBJECT
                        n = 54, #number of samples
                        cexRow = 0.8, #font size y axis
                        cexCol = 0.4) #font size x axis
plotCorr(obj = obj,
                n = 200, #number of microorganisms plotted
                cexRow = 0.25,
                cexCol = 0.25,
                col = heatmapCols)
png(filename="MicrobiomeCorr_PNG.png", height=5000, width=5000, res=300)
plotCorr(obj = obj, n = 77, cexRow = 0.4, cexCol = 0.4, col = heatmapCols)
dev.off()
```

The sample versus microbe matrix is plotted with plotMRheatmap() function, while the microbe correlation matrix is plotted by the plotCorr() function.

4.2.3 Pavian (WorkInProgress)

https://github.com/fbreitwieser/pavian RStudio tool to visualise and compare Kraken2 reports (others also possible). Requires –report argument in the kraken2 command.

```
pavian::runApp(port=5000, maxUploadSize=500*1024^2)
```

4.3 De Novo Assemblies

The generation of an assembly works most efficiently if at least a partial overlap between reads is present, preferably paired-end reads (e.g. unstacked paired-end data of shotgun sequencing). DArTseq generates stacked gapped single-end read data, and is therefore not optimal at making assemblies.

4.4 The Endogenous Banana Streak Virus (WorkInProgress)

To detect the presence of an endogenous virus, only the reads mapped to the reference genome need to be taken into account. And this only if the reference genome is known to contain endogenous virus sequences.

```
Detection of eBSV in balbisiana
(1) extract mapped reads, discard unmapped reads (indication of provirus)
samtools idxstats in.bam | cut -f1 | grep ‘Bscaffold\|Unmapped’ | xargs samtools view -o out
(2) convert the bam to fastq
samtools fastq in.bam > out.fastq.gz
(3) BWA mem new fastq file to eBSV reference genomes
bwa mem -r ref.fasta in.fastq.gz > out.sam
(4) extract mapped reads only
→ samtools view -F 4 INPUT.sam > OUTPUT.sam
grep -v ^@ INPUT.sam | awk '{print "@"$1"\n"$10"\n+\n"$11}' > OUTPUT.fastq
```

eBSV Strain Isolates 11 isolates of Banana Streak Virus NCBI BioProject: PRJNA485481 Map DArTseq reads with BWA MEM on the 11 isolates as reference genomes James, A. P., Geijskes, R. J., Dale, J. L., Harding, R. M. (2011). Molecular characterisation of six badnavirus species associated with leaf streak disease of banana in East Africa. Annals of Applied Biology, 158(3), 346–353. doi:10.1111/j.1744-7348.2011.00466.x Harper et al. (2005) reported the presence of 13 distinct BSV sequence groups from Uganda, named consecutively as Banana streak Uganda A virus to Banana streak Uganda M virus. Banana streak acuminata Yunnan virus, has been deposited in the NCBI database (GenBank accessions DQ092436), phylogenetically most closely related to BSVNV (Gayral Iskra-Caruana, 2009). Geering et al. (2000, 2005a) reported partial sequences of two BSV isolates from Australia [named Banana streak Cavendish virus (BSV-Cav) and Banana streak Imove virus (BSIMV)] Gayral P, Noa-Carrazana JC, Lescot M, Lheureux F, Lockhart BE, Matsumoto T, Piffanelli P, Iskra-Caruana ML. A single Banana streak virus integration event in the banana genome as the origin of infectious endogenous pararetrovirus. J Virol. 2008 Jul;82(13):6697-710. doi: 10.1128/JVI.00212-08. Epub 2008 Apr 16. PMID: 18417582; PMCID: PMC2447048. BSV Goldfinger (BSGfV) present in the wild diploid M. balbisiana cv. Pisang Klutuk Wulung (PKW) Muller, E., Ullah, I., Dunwell, J.M. et al. Identification and distribution of novel badnaviral sequences integrated in the genome of cacao (Theobroma cacao). Sci Rep 11, 8270 (2021). https://doi.org/10.1038/s41598-021-87690-1

```
AcuminataVietnam (Vietnam)
AcuminataYunnan (Yunnan, China)
Cavendish (CA; Australia)
Imove (IM; Australia)
Mysore (MY; Tonga)
GoldFinger (GF; PKW, Indonesia)
(OL; Nigeria)
U(A/I/L/M; Uganda)
```

5 BEAUti and the BEAST

BEAST and attached programs have interactive user interfaces available on Windows. Links for download:
download BEAST, http://www.beast2.org/
download TRACER, https://github.com/beast-dev/tracer/tree/v1.7.2
download FIGTREE, https://github.com/rambaut/figtree

5.1 BEAUti XML File

First the merged VCF file needs to be converted into NEXUS format. For this, download the vcf2phylip.py script from its github repository. For binary NEXUS files, use the -b argument instead of -n.

Ubuntu terminal

```
# download script and place it somewhere convenient
git clone https://github.com/edgardomortiz/vcf2phylip
# script command
python3 vcf2phylip.py -i VCF --output-folder . --output-prefix PREFIX -n -p
```

The BEAUTi program can import the NEXUS file and transform it into a XML file. Plenty of options can be chosen:

Tab	**Options**
Partitions	Info about imported data set.
Tip Dates	Enable 'Use tip dates' to fill in the dates of each sample manually or import from a file (not useful for contemporary samples).
Site Model	The evolutionary model settings for BEAST. The options available depend on whether the data are nucleotides, or amino acids, binary data, or general data.
Clock Model	Select the molecular clock model.
Priors	Allows priors to be specified for each parameter in the model. The model selections made in the Site Model and Clock Model tabs, result in the inclusion of various parameters in the model, and these are shown in the priors tab. Default settings are robust beginner settings.
MCMC	General settings to control the length of the MCMC run and the file names. Log = iterations/1000

5.2 BEAST2 Analysis

Now run BEAST and when it asks for an input file, provide your newly created XML file as input. BEAST will then run until it has finished. The actual results files are saved to the same location as your input file. Multiple files can be combined with logCombiner.

5.3 LogCombiner

Logs of multiple runs can be combined with logCombiner.

5.4 Tracer

Detailed view of the logs.

5.5 TreeAnnotator

Build the tree.

5.6 FigTree

FIGTREE visualises consensus tree

References

Bouckaert, R., Vaughan, T.G., Barido-Sottani, J., Duchêne, S., Fourment, M., Gavryushkina, A., Heled, J., Jones, G., Kühnert, D., De Maio, N., Matschiner, M., Mendes, F.K., Müller, N.F., Ogilvie, H.A., du Plessis, L., Popinga, A., Rambaut, A., Rasmussen, D., Siveroni, I., Suchard, M.A., Wu, C., Xie, D., Zhang, C., Stadler, T., Drummond, A.J. **(2019)** BEAST 2.5: An advanced software platform for Bayesian evolutionary analysis. *PLoS computational biology.* **15**(4): e1006650. doi.org/10.1371/journal.pcbi.1006650

6 Functional Proteomics

Functional proteomics aims at the elucidation of the biological function of unknown proteins and the definition of cellular mechanisms at the molecular level. However, here we investigate the effect of SNPs within known proteins by looking at the gene, the CDS, and the exon/intron distinction.

Gene Families For the analyses explained in this section, we start with the assumption a set of genes of interest and their CDS is available. Many genes and their CDS are available online in multiple databases. Examples:
`http://planttfdb.gao-lab.org`
`https://www.ncbi.nlm.nih.gov/`
`http://pdgd.njau.edu.cn:8080/`

6.1 Variant Calling

For the discovery of SNPs within the genes of interest, we use a similar pipeline (sequence of tools) as described in **1.2.5 DArTseq Analysis**. First, we assume all the genes of interest are in seperate fasta files in a folder `Genes` and all the corresponding CDS are also in seperate fasta files in a folder `GenesCDS`. If you have concatenated fasta files, call them `Genes.fasta` and `GenesCDS.fasta` respectively. Use the script below to split all genes and corresponding CDS into separate fasta files. Also, give the CDS sequences the same filename as the genes, but followed with "CDS". See the table for an example.

Full Gene Name	Coding Sequence Name
Gene1.fasta	Gene1CDS.fasta
Putative_protein.fasta	Putative_proteinCDS.fasta

Bash script

```
#!/bin/bash
cat Genes.fasta | awk '{
        if (substr($0, 1, 1)==">") {filename=(substr($0,2) ".fasta")}
        print $0 >> filename
        close(filename)
}'
cat GenesCDS.fasta | awk '{
        if (substr($0, 1, 1)==">") {filename=(substr($0,2) ".fasta")}
        print $0 >> filename
        close(filename)
}'
```

First, generate locations of all CDS on all genes. The `-splitD` argument is necessary to keep gaps (=introns) from the mapping step and recognize the separate exon blocks. The mapping is not flawless and more often than not contains faults. Use tools like Geneious Prime or Muscle to align the CDS on the gene and check the coordinates in the BED file are correct.

```bash
#!/bin/bash
# Index all full genes seperately to use as reference
for file in ./Genes/*fasta; do
   bwa index $file
   samtools faidx $file
   java -jar picard.jar CreateSequenceDictionary -R $file
done
# Map eahc CDS on the corresponding gene
for file in ./GenesCDS/*CDS.fasta; do
    filename=$(basename -- "$file")
    bwa mem ./Genes/"${filename%CDS.fasta}.fasta" $file > "${filename%.fasta}.sam"
done
# SAM to BAM with picard
for file in ./*CDS.sam; do
    java -jar picard.jar AddOrReplaceReadGroups -I $file -O "${file%.sam}.bam" \
        --CREATE_INDEX -SO coordinate -RGLB "${file%.sam}" -RGPL illumina \
        -RGPU run -RGSM "${file%.sam}" -RGID "${file%.sam}"
done
# Convert BAM to BED, keep gaps with -splitD
for file in ./*CDS.bam; do
        bedtools bamtobed -splitD -i $file > "${file%.bam}.bed"
done
# Concatenate all BEDs into one
cat *.bed > GenesCDS.bed
```

Secondly, call variants of DArTseq samples with all the genes as reference. Index the full genes set as reference before variant calling. The sequence of commands/tools is similar to **1.2.5 DArTseq Analysis**. Start with the processed reads after the GBprocesS step (sample.GbprocesS.fastq.gz).

```bash
#!/bin/bash
# Index the complete set of genes as reference
bwa index genes.fasta
samtools faidx genes.fasta
java -jar picard.jar CreateSequenceDictionary -R genes.fasta
```

```bash
#!/bin/bash
# Map all samples on all genes
for file in ./*GBprocesS.fastq.gz; do
    bwa mem Genes.fasta $file > "${file%.GBprocesS.fastq.gz}.sam"
done
# Sort with picard
for file in ./*sam; do
    java -jar picard.jar AddOrReplaceReadGroups -I $file -O "${file%.sam}.bam" \
        --CREATE_INDEX -SO coordinate -RGLB "${file%.sam}" -RGPL illumina \
        -RGPU run -RGSM "${file%.sam}" -RGID "${file%.sam}"
done
# Call variants with FreeBayes
for file in ./*bam; do
    freebayes -f Genes.fasta --genotype-qualities --strict-vcf \
        --vcf "${file%.bam}.vcf" $file
done
# Filter SNPs
for file in ./*vcf; do
    gatk SelectVariants -R Genes.fasta -V $file --select-type-to-include SNP \
        -O "${file%.vcf}.snp.vcf"
done
```

6.2 KaKsCalculator 3.0

The ratio of Ka and Ks represents the nonsynonymous/synonymous substitution rate and also calculates the selective pressure on (non-)coding sequences. The third version of the KaKs_Calculator tool is available at https://ngdc.cncb.ac.cn/biocode/tools/BT000001/releases/3.0 [39]. The KaKs analysis requires CLUSTALW alignment as input. Explained below is the installation and the scrip for generating alignments for each CDS of interest across all samples.

```
# Install EMBOSS and muscle
sudo apt install emboss
sudo apt install muscle
# Download KaKs_Calculator3.0.zip
# Install KaKs_Calculator3.0
mv /mnt/c/Users/USER/Downloads/KaKs_Calculator3.0.zip /home/USER/bin/
cd /home/USER/bin/
unzip KaKs_Calculator3.0.zip
cd KaKs_Calculator/src/
make
# Installation in /home/USER/bin/KaKs_Calculator/src
# Add to $PATH in .bashrc
nano /home/USER/.bashrc
```

The SNPs for each sample in its VCF file are transformed into fasta sequences which can be added in a CLUSTALW alignment. This is the direct input for the KaKs_Calculator command.

```
#!/bin/bash
# Create consensus sequence for each gene and sample
for file in ./*snp.vcf; do
    bgzip $file
```

```
done
for file in ./*snp.vcf.gz; do
    tabix -f -h $file
    cat Genes.fasta | bcftools consensus $file > "${file%.snp.vcf.gz}.genes.fasta"
done
# Filter out CDS for each sample and merge into one sequence
# With union command from EMBOSS
for file in ./*genes.fasta; do
    echo $file
    for bedfile in ./GenesCDS/*bed; do
        bedname=$(basename -- "$bedfile")
        bedprefix=$(basename $bedname .bed)
        echo $bedprefix
        if [ -d ./Alignment/"$bedprefix" ]; then
            bedtools getfasta -fi $file -bed $bedfile -split \
                -fo ./Alignment/"$bedprefix"/"${file%.genes.fasta}.$bedprefix.fasta"
        else
            mkdir ./Alignment/$bedprefix
            bedtools getfasta -fi $file -bed $bedfile -split \
                -fo ./Alignment/"$bedprefix"/"${file%.genes.fasta}.$bedprefix.fasta"
        fi
    done
done
for file in ./Alignment/*/*fasta; do
    filename=$(basename -- "$file")
    fileprefix=$(basename $filename .fasta)
    echo $fileprefix
    mv $file temp.fasta
    union -osdbname2 $fileprefix -filter temp.fasta > $file
done
# Make CLUSTALW alignment with MUSCLE
for dir in ./Alignment/Ma*; do
    dirname=$(basename -- "$dir")
    cat ./Alignment/$dirname/*CDS.fasta > ./Alignment/$dirname/"$dirname"_FASTA.fasta
done
for file in ./Alignment/Ma*/*FASTA.fasta; do
    muscle -in $file -clwstrict -out "${file%FASTA.fasta}CLUSTAL.aln"
done
# Convert to AXT with AXTConvertor from KaKsCalculator3.0
for file in ./Alignment/Ma*/*CLUSTAL.aln; do
    AXTConvertor $file "${file%CLUSTAL.aln}AXT.axt"
done
# Calculate KaKs for every AXT alignment
for file in ./Alignment/Ma*/*AXT.axt; do
```

```
    KaKs -i $file -o "${file%_AXT.axt}.kaks"
done
```

When KaKsCaluclator3.0 gives error messages like [Error. The sequences are not codon-based alignment.], then the mapping of the CDS on the gene contains a fault. Perform, for example, a Muscle alignment of the CDS and the corresponding gene in Geneious Prime. Check if the aligned areas correspond with the coordinates given in the generated BED file. Adjust if necessary.

6.3 Annotate Variants

6.3.1 ANNOVAR

ANNOVAR is a software tool to functionally annotate genetic variants [40]. The gene-based annotation identifies whether SNPs cause protein coding changes and the amino acids that are affected. Download ANNOVAR by registering an email address on the ANNOVAR website (https://www.openbioinformatics.org/annovar/annovar_download_form.php) and clicking the download link for the ANNOVAR package (latest version) when the email arrives. Use the instructions provided if the email suffix is not recognized (not yet added in the authorization list). Unpacking the package will result in an annovar folder containing all the required scripts.

```
# Move the package into /home/USER/bin
mv /mnt/c/Users/USER/Downloads/annovar.latest.tar.gz /home/USER/bin/
cd /home/USER/bin/
tar -xvfz annovar.latest.tar.gz
# Scripts in the directory /home/USER/bin/annovar/
```

ANNOVAR requires a GFF3 input to specify the CDS regions within the gene. Here, GMAP [41] is used to generate a GFF3 file from the BED file and the fasta sequence files. We make a GMAP database directory GenesDB for the set of genes of interest. For some reason, the capital D in Dir= causes errors later on. Therefore we utilize the sed funcitonality to replace Dir= with dir= in the generated GFF3 file.

```
# Install GMAP
sudo apt-get install gmap
# Additional commands in /usr/lib/gmap
# Add to $PATH in .bashrc file
nano /home/USER/.bashrc
# Make GMAP database directory anywhere and build reference
mkdir GenesDB
gmap_build -D GenesDB -d Genes Genes.fasta
# Make MAP file of the CDS and convert to GFF3
gmap -D GenesDB -d Genes -f 7 GenesCDS.fasta > GenesCDS.map
cat GenesCDS.map | iit_store -o GenesCDS
gmap -D GenesDB -d Genes -M . -m GenesCDS -f 2 GenesCDS.fasta > GenesCDS.gff3
# Replace Dir with dir
sed -i 's/Dir=/dir=/g' GenesCDS.gff3
```

For the full protocol and sequence of commands for gene-based annotation on a VCF file, check Yang and Wang (2015) [42]. Here, an abbreviated version is shown, also using the `gff3ToGenePred` script available at `http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/`

```
# Convert to GenePred
gff3ToGenePred GenesCDS.gff3 GenesCDS_refGene0.txt
# Adding line number
nl GenesCDS_refGene0.txt > GenesCDS_refGene.txt
# Retrieve CDS fasta sequence
retrieve_seq_from_fasta.pl -format refGene -seqfile Genes.fasta \
    -out GenesCDS_refGeneMrna.fa GenesCDS_refGene.txt
# Annotate VCF file
table_annovar.pl VCF.snp.vcf . -vcfinput -buildver GenesCDS -protocol refGene \
    -out VCF.annovar -operation g
```

6.3.2 Variant Effect Predictor

Variant Effect Predictor (VEP) from Ensembl determines the effect of variants on genes, transcripts, and protein sequences, as well as regulatory regions [43].

```
# Install dependencies
sudo apt-get install perl libdbi-perl libdbd-mysql-perl libbio-db-hts-perl bioperl
sudo apt install cpanminus libmysqlclient-dev
# Make directory for cpanm modules
mkdir -p $HOME/cpanm
# Add to .bashrc from home directory
# export PERL5LIB="$PERL5LIB:$HOME/cpanm/lib/perl5"
nano /home/USER/.bashrc
# Install cpanm modules
cpanm -l $HOME/cpanm Set::IntervalTree
cpanm -l $HOME/cpanm Archive::Zip
cpanm -l $HOME/cpanm DBD::mysql
cpanm -l $HOME/cpanm DBI
cpanm -l $HOME/cpanm Bio::DB::HTS
# Install VEP
cd /home/USER/bin/
git clone https://github.com/Ensembl/ensembl-vep.git
cd ensembl-vep
perl INSTALL.pl --NO_HTSLIB
# Add ensembl-vep directory to .bashrc from home directory
# export PATH:"/usr/bin:/home/USER/bin/ensembl-vep:$PATH"
nano /home/USER/.bashrc
```

VEP needs some input containing variant positions to run, provided in each sample VCF file. VEP can integrate custom annotation from standard format files, here the `-gff` flag for the `GenesCDS.gff3` file generated in the **6.3.1**

ANNOVAR section. Custom annotation files must first be prepared in a particular way in order to work with `tabix` and therefore with VEP. Files must be stripped of comment lines, sorted in chromosome and position order, compressed using `bgzip` and finally indexed using `tabix`, as shown below.

```
# GFF3 preparation
grep -v "#" GenesCDS.gff3 | sort -k1,1 -k4,4n -k5,5n -t$'\t' | \
    bgzip -c > GenesCDS.gff3.gz
# Tabix the GFF3 file
tabix -p gff GenesCDS.gff3.gz
# VEP command
vep -i VCF.snp.vcf --gff GenesCDS.gff3.gz --fasta Genes.fasta --vcf -o VCF.vep.vcf
```

7 fastStructure

The program Structure uses multi-locus genotype data to investigate population structure, including inferring the presence of distinct populations, assigning individuals to populations, studying hybrid zones, identifying migrants and admixed individuals, and estimating population allele frequencies in situations where many individuals are migrants or admixed. It can be applied to most of the commonly-used genetic markers, including SNPS, microsatellites, RFLPs and AFLPs. The online program Structure Harvester provides tools for visualising Structure output and for determining K, but does not work with fastStructure output. The tool fastStructure is built on Structure for running very large SNP data sets, exactly what the DArTseq pipeline generates.

The installation of fastStructure is more complex than other tools. Below is a link to a video explaining the installation process while referring to the installation guide by Josiah Altschuler (also below). The adapted installation guide in full is below, with additional steps for creating a virtual python2 environment.

Youtube Tutorial: `https://www.youtube.com/watch?v=tAScTlKW60w`

Installation Guide: `https://gist.github.com/josiahaltschuler/a063e03b4197013def53f9a0abc6dfed`

Ubuntu terminal

```
# replace USERNAME with ubuntu account name!
# update system and install packages
sudo apt update
sudo apt upgrade
sudo add-apt-repository universe
sudo apt install python2 virtualenv
# create and activate python2 virtual environment
cd
virtualenv --python=$(which python2) venv/fastStructure/
source venv/fastStructure/bin/activate #terminate with "deactivate"
# install dependencies and specify version
pip install numpy==1.16.5
pip install cython==0.27.3
pip install scipy==1.2.1
mkdir /home/USERNAME/bin
cd /home/USERNAME/bin
# download and install latest gsl
wget ftp://ftp.gnu.org/gnu/gsl/gsl-2.7.tar.gz
tar -xf gsl-2.7.tar.gz
cd gsl-2.7
./configure --prefix=/home/USERNAME/bin/gsl-2.7
make
make check
make install
# in .bashrc add, or update exports
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/USERNAME/bin/gsl-2.7/lib
export CFLAGS="-I/home/USERNAME/bin/gsl-2.7/include"
export LDFLAGS="-L/home/USERNAME/bin/gsl-2.7/lib"
```

```
# close and reopen terminal to refresh new .bashrc settings
sudo apt-get install python-dev
# close and reopen terminal to refresh new global settings
# install fastStructure
source venv/fastStructure/bin/activate
cd /home/USERNAME/bin/
wget --no-check-certificate https://github.com/rajanil/fastStructure/archive/master.tar.gz
tar -xf master.tar.gz
mv fastStructure-master/ fastStructure
cd fastStructure/vars
python setup.py build_ext -f --inplace
cd ..
python setup.py build_ext -f --inplace
```

Before running fastStructure on a multi-sample VCF file, the format needs to be converted to Structure, which is also the input for fastStructure. For this, use PGDSpider available on the following website: `http://www.cmpg.unibe.ch/software/PGDSpider/`. After conversion, the file can be analysed by fastStructure in the python2 environment. The –input= argument requires only the prefix of the file, the extension is specified in the –format= option. The same applies to the –output= argument, only give the desired output prefix.

```
# check if the python2 environment is active!
(python2env)(base)sander@sander:~$
#  command with K=3
python structure.py -K 3 --input=Musa --output=Musa_fS --format=str
```

The generated .meanQ file can be used as the direct input for the distruct visualisation tool. The distruct.py script comes with the fastStructure installation, but the modified version by Vikram E. Chhatre (available at https://vc.popgen.org/software/distruct/) offers an additional parameter for the graphical output. Simply register and download, and unpack the python scripts (also to be run in the python2 environment).

Whenever encountering the "error _tkinter.TclError: no display name and no $DISPLAY environment variable" error, open the distruct script in a text editor and add "import matplotlib" and "matplotlib.use('Agg')" in seperate lines before the "import matplotlib.pyplot as plot" line.

Whenever encountering the "IndexError: tuple index out of range" error, open the distruct script in a text editor and either manually add more colours (at least as many as the K in the command; default 3 colours), or switch to automatically assigning colours by (de)commenting some lines.

```
# check if the python2 environment is active!
(python2env)(base)sander@sander:~$
# distruct command with K=3
python distruct2.2.py -K 3 --input=Musa_fS --output=Musa_fS_d \
        --popfile=popfile.txt --poporder=poporder.txt \
        --title="Musa with K=3"
```

Another option for visualising the fastStructure results is the online tool CLUMPAK. It accepts a .zip archive of the .meanQ file that fastStructure generates and produces a bar graph like the distruct python script. CLUMPAK is available at http://clumpak.tau.ac.il/index.html. To automate the analysis of fastStructure for an amount of different K-values, use the script below. Simply specify the highest value for K as the ‘END’ variable, and the script will run every value up to ‘END’.

Bash script

```bash
#!/bin/bash
END=10
for i in $(seq 1 $END); do
        echo "fastStructure analysis for K=$i on provided structure file"
        python structure.py -K $i --input=IN --output=OUT_fS --format=str
done
```

References

Earl, D.A., vonHoldt, B.M. **(2012)** STRUCTURE HARVESTER: a website and program for visualizing STRUCTURE output and implementing the Evanno method. *Conservation Genetics Resources.* **4**:(2): 359-361. doi.org/10.1007/s12686-011-9548-7

Kopelman, N.M., Mayzel, J., Jakobsson, M., Rosenberg, N.A., Mayrose, I. **(2015)** CLUMPAK: a program for identifying clustering modes and packaging population structure inferences across K. *Molecular Ecology Resources.* **15**(5): 1179-1191. doi.org/10.1111/1755-0998.12387

Lischer, H.E.L., Excoffier, L. **(2012)** PGDSpider: An automated data conversion tool for connecting population genetics and genomics programs. *Bioinformatics.* **28**: 298-299. doi.org/10.1093/bioinformatics/btr642

Raj, A., Stephens, M., Pritchard, J.K. **(2014)** fastSTRUCTURE: Variational Inference of Population Structure in Large SNP Data Sets. *Genetics.* **197**(2): 573-589. doi.org/10.1534/genetics.114.164350

8 Principal Components

8.1 Principal Component Analysis

A principal component analysis (PCA) computes the principal components and uses them to perform a change of basis on the data, only using the first principal components and ignoring the rest. In population genetics, it is a statistical method commonly used to identify structure in the distribution of genetic variation mainly across geographical locations. **Unlike phylogenetics, PCAs are quite resistant against missing data.** The first step is using PLINK1.9 to generate a distance matrix of the multi-sample VCF file. The matrix can be further analysed in RStudio. The use of –double-id and –allow-extra-chr depends on the names of the chromosomes of the reference used in the DArTseq pipeline. The default is "chr" and Musa balbisiana, for example, uses "Bchr". The output are .mdist and .mdist.id files with the prefix specified in the –out argument.

Ubuntu terminal

```
plink --vcf VCF --double-id --allow-extra-chr --distance-matrix --out VCF_DM
```

RStudio script

```
# load distance matrix, pop IDs, and ind IDs
dist_populations<-read.table("VCF_DM.mdist",header=F)
pop <- data.frame(pop=read.table("VCF_DM.mdist.id")[,1])
popInd <- data.frame(popInd=read.table("VCF_DM.mdist.id")[,2])
# generate eigenvectors and PCA
mds_populations <- cmdscale(dist_populations,eig=T,5)
eigenvec_populations <- cbind(pop,popInd,mds_populations$points)
eigen_percent <- round(((mds_populations$eig)/sum(mds_populations$eig))*100,2)
# PCA plot
library(ggforce)
library(tidyverse)
ggplot(data = eigenvec_populations, aes(x=`1`, y=`2`, label=popInd)) +
     geom_point(mapping=aes(x=`1`, y=`2`, color=pop), show.legend=TRUE) +
     geom_text(hjust="inward", vjust="inward", size=1.5) +
     geom_mark_ellipse(aes(color=pop, fill=pop, filter=pop), show.legend=FALSE) +
     geom_hline(yintercept=0, linetype="dotted") +
     geom_vline(xintercept=0, linetype="dotted") +
     labs(title = "PCA of VCF",
          x = paste0("PC1 (",eigen_percent[1]," %)"),
          y = paste0("PC2 (",eigen_percent[2]," %)")) +
     theme_minimal()
```

In the same way gene-based phylogenetic analyses are based on a portion of the variant data, PCA analyses can be performed on a subset of variants. For this, the variants in the VCF files need to be filtered based on genomic coordinates before further analysis. This is described in the Gene-based Phylogenetics section.

8.2 Discriminant Analysis

Discriminant analysis of principal components (DAPC).

9 Hybridization

9.1 SMAP

Natural hybridization between genetically different populations or taxa results in gene flow between them. The occurrence and degree of hybridization can be investigated by looking into the stack mapping anchor points (SMAPs) with the tool by the same name. The manual and detailed explanation can be found here: https://ngs-smap.readthedocs.io/en/latest.

Ubuntu terminal

```
# SMAP installation
pip install --upgrade pip
pip install ngs-smap
```

The multi-sample VCF file needs to be filtered to guarantee the genotype calls are supported by sufficient raw reads. The same filtration commands as under the Phylogenetics section may be used to remove non-variant calls and multi-allelic SNPs, and the use of FilterVCF.py. As well as the additional command with VCFtools. The arguments are as follows; –mac only includes sites with minor allele count equal or greater than 2; –minDP specifies the required minimum number of reads supporting each of the reported alleles; –minGQ signifies the Phred-scaled confidence score for genotype call; –minQ includes only sites with quality above this threshold. This VCFtools command may potentially have no influence on the VCF file.

```
vcftools --vcf VCF --mac 2 --minDP 20 --minGQ 20 --minQ 20 --recode --out VCF
```

The BAM files corresponding to the samples in the multi-sample VCF file are to be put in one directory. For simplicity's sake, the directory BAM/ holds all the BAM files. The smap delineate command generates a .bed file necessary for the smap haplotype-sites command later on. Details on the other arguments can be found in the manual mentioned above.

```
#generate .bed file
smap delineate BAMs/ -r stranded -p 2 -q 20 -x 5 -y 1500 -w 5 -c 10 -s 20
smap haplotype-sites BAMs/ BED VCF -r stranded -a include --plot all -o OUT
smap haplotype-sites BAMS/ BED VCF -r stranded -a include --no_indels -q 20 -c 10 -f 5 -e do            10
```

For generating Jaccard graphs and other visualisation, the python3 SMAPapp-Matrix.py is used. This script is available at https://github.com/sanderdebacker/DArTseq$_p$*ipelineinthefoldersmapapps−main.Itusesthehaplotypes*$_d$*iscrete*$_c$*alls* *sitescommandabove.The−−plot*$_l$*ine*$_c$*urvescanfurtherbespecifiedbyalsousingthe−−list*$_l$*ine*$_c$*urvesargument.Possibly,themodu* *nargumentcanbeusedtoorderthesamplesinthematrixoutputintogroups.*

```
#install natsort module
pip install natsort
#SMAPapp-matrix command for line curves
python3 SMAPapp-matrix.py --table TSV -p 2 --plot_line_curves -lc 0.05
#SMAPapp-matrix command for matrices
```

```
python3 SMAPapp-matrix.py --table TSV -p 2 -lc 0.05 -sc 2500 --print_sample_information All
```

SNP filtering DArTseq pipeline of banksiis and schizos and hybrids with Musa acuminata Pahang as reference Merge banksiis and remove SNPs that have all same ALT (1/1 non-variants) Make bed file of remaining SNPs Merge schizos and remove SNPs that have all same ALT (1/1 non-variants) Make bed file of remaining SNPs Intersect in both ways: Only SNPs that are exclusive in schizos → schizo.bed Only SNPs that are exclusive in banksiis → banksii.bed Filter hybrids with schizo.bed → schizo oriented clustering Filter hybrids with banksii.bed → banksii oriented clustering

9.2 Ancestral Chromosome Painting

This section is aimed at studying the ancestral contribution along chromosomes of one accession or a group of accessions. Based on the vcf file provided and a file with the names of ancestral accessions, the origin of each allele is attributed to an ancestral group. Download picard.jar and GATK3.7 , and samtools and bamtools as described above.

Ubuntu terminal

```
#installation STAR (/home/USER/STAR/source/STAR) | ONLY FOR RNA
git clone https://github.com/alexdobin/STAR.git
cd STAR/source
make STAR
#installation bam-readcount (/home/USER/bam-readcount/build/bin/bam-readcount)
sudo apt install cmake
git clone https://github.com/genome/bam-readcount
cd bam-readcount
mkdir build
cd build
cmake ..
make
#installation GNUplot (/usr/bin/gnuplot)
sudo apt-get install gnuplot
#installation Circos (/home/USER/circos/bin)
#download latest version from http://circos.ca/software/download/
mv /mnt/c/Users/USER/Downloads/circos-current.tgz .
tar -xvf circos-current.tgz
mv circos-0.xx.x/ circos/
#installation VcfHunter (scripts in ./VcfHunter/bin/)
git clone https://github.com/SouthGreenPlatform/VcfHunter
```

The loca_programs.conf file contains the installation locations of the different tools required for the different analyses available in VcfHunter. It should look something like:

Configuration file

```
[Programs]
star = /home/USER/STAR/source/STAR
picard = /home/USER/picard/picard.jar
java = /usr/bin/java
gatk = /home/USER/GATK/GATK3.7/GenomeAnalysisTK.jar
samtools = /home/USER/samtools/samtools
plotbamstats = /home/USER/samtools/misc/plot-bamstats
bamtools = /usr/bin/bamtools
bwa = /usr/bin/bwa
python = python
bamreadcount = /home/USER/bam-readcount/build/bin/bam-readcount
circos = circos
```

9.3 Problem Solving - Encountered Errors

The original scripts contain references to modules not available anymore in the most recent versions of Bio.python. Also, the use of outdated versions of tools requires the use of other versions of dependencies. When receiving one of the following error messages, perform the accompanying instructions.

Spline *ImportError: cannot import name ’spline’ from ’scipy.interpolate’*
The module *spline* was removed in newer versions of *scipy*. Open the python script with which this error occurred.

```
# replace
from scipy.interpolate import spline
# with
from scipy.interpolate import make_interp_spline
```

UnifiedGenotyper *ERROR MESSAGE: Invalid command line: Malformed walker argument: Could not find walker with name: UnifiedGenotyper*
This means the wrong version of java environment is being used. Install the right version or change to an alternative if already installed.

```
# install java run environment 8
sudo apt install openjdk-8-jre*
# change java version
sudo update-alternatives --config java
```

Rectangle Object *AttributeError: ’Rectangle’ object has no property ’normed’*

```
# in the python script, replace every instance of
normed=1
# with
density=True
```

BWA Read Group *Line : 677 - Error in step A (bwa-mem): [E::bwa_set_rg] the read group line contained literal <tab> characters – replace with escaped tabs: \t*

This error can occur in the process_reseq.1.0.py script. Replace line 657 in the /utilsSR/utilsSR.py script.

```
# original
rg_tag = '@RG"\t"ID:'+ACC_ID+'"\t"SM:'+ACC_ID+'"\t"LB:'+ACC_ID+
        '"\t"PU:whatever"\t"PL:ILLUMINA'
# replace with
rg_tag = '@RG"\\t"ID:'+ACC_ID+'"\\t"SM:'+ACC_ID+'"\\t"LB:'+ACC_ID+
        '"\\t"PU:whatever"\\t"PL:ILLUMINA'
```

LibQt5Core gnuplot: error while loading shared libraries: libQt5Core.so.5: cannot open shared object file: No such file or directory Depending on the location of the library, a different command will solve this problem.

```
# check installation
sudo apt-get install libqt5gui5
# two examples of library locations
strip --remove-section=.note.ABI-tag /usr/lib/x86_64-linux-gnu/libQt5Core.so.5
strip --remove-section=.note.ABI-tag /usr/lib64/libQt5Core.so.5
```

For each ancestral genome multiple non admixed ancestral accessions are possible. schizo1 SS banksii1 AA banksii2 AA balbisiana1 BB

DArTseq_pipeline_VcfHunter → process_reseq_1.0.py –conf reseq_conf.txt –steps b,c,d,e,f,g → vcf file generated for input in vcf2allPropAndCov.py –vcf VCF –origin Origin.txt –acc SAMPLE –ploidy 2 !!step e of process_reseq_1.0.py starts with _real_recal.bam (can name general bam from DArTseq_pipeline_VcfHunter SAMPLE_real_recal.bam and continue??)!! !!Maybe naming _real_recal.bam can be a direct input for step e??!! Malaccensis - acuminata AA Burmannica - acuminata AA Siamea - acuminata AA Zebrina - acuminata AA Balbisiana - balbisiana BB Schizocarpa - schizocarpa SS Peekelii - australimusa UU Lolodensis - australimusa UU Maclayi - australimusa UU Boman - australimusa UU

References

[1] D. Jaccoud, K. Peng, D. Feinstein, and A. Kilian. Diversity Arrays: a solid state technology for sequence information independent genotyping. *Nucleic Acids Research*, 29(4):e25, 2001. doi: 10.1093/nar/29.4.e25.

[2] A. Risterucci, I. Hippolyte, X. Perrier, L. Xia, V. Caig, M. Evers, E. Huttner, A. Kilian, and J. Glaszmann. Development and assessment of Di-

versity Arrays Technology for high-throughput DNA analyses in Musa. *Theoretical and Applied Genetics.*, 119(6):1093–1103, 2009. doi: 10.1007/s00122-009-1111-5.

[3] A. Kilian, P. Wenzl, E. Huttner, J. Carling, L. Xia, H. Blois, V. Caig, K. Heller-Uszynska, D. Jaccoud, C. Hopper, M. Aschenbrenner-Kilian, M. Evers, K. Peng, C. Cayla, P. Hok, and G. Uszynski. Diversity Arrays Technology: A Generic Genome Profiling Technology on Open Platforms. *Data Production and Analysis in Population Genomics*, 888(Chapter 5): 67–89, 2012. doi: 10.1007/978-1-61779-870-2_5.

[4] P. Danecek, J.K. Bonfield, J. Liddle, J. Marshall, V. Ohan, M.O. Pollard, A. Whitwham, T. Keane, S.A. McCarthy, R.M. Davies, and H. Li. Twelve years of SAMtools and BCFtools. *Gigascience*, 10(2):giab008, 2021. doi: 10.1093/gigascience/giab008.

[5] E. Garrison and G. Marth. Haplotype-based variant detection from short-read sequencing. *arXiv preprint*, arXiv(q-bio.GN):1207.3907, 2012. doi: 10.48550/arXiv.1207.3907.

[6] Broad Institute. Picard Toolkit. *Broad Institute, GitHub repository*, Available at broadinstitute.github.io/picard/, 2019.

[7] G.A. Van der Auwera, M. Carneiro, C. Hartl, R. Poplin, G. del Angel, A. Levy-Moonshine, T. Jordan, K. Shakir, D. Roazen, J. Thibault, E. Banks, K. Garimella, D. Altshuler, S. Gabriel, and M. DePristo. From FastQ Data to High-Confidence Variant Calls: The Genome Analysis Toolkit Best Practices Pipeline. *Current Protocols in Bioinformatics*, 43: 11.10.1–11.10.33, 2013. doi: 10.1002/0471250953.bi1110s43.

[8] D. Schaumont. GBprocesS: Genotyping-by-Sequencing Data Processing Toolkit. *[Online]*, Available at gitlab.com/dschaumont/GBprocesS, 2020.

[9] P. Danecek, A. Auton, G. Abecasis, C.A. Albers, E. Banks, M.A. DePristo, R. Handsaker, G. Lunter, G. Marth, S.T. Sherry, G. McVean, R. Durbin, and 1000 Genomes Project Analysis Group. The Variant Call Format and VCFtools. *Bioinformatics*, 27:2156–2158, 2011. doi: 10.1093/bioinformatics/btr330.

[10] J. Goudet. Hierfstat, a package for R to compute and test hierarchical F-statistics. *Molecular Ecology Notes*, 5:184–186, 2005. doi: 10.1111/j.1471-8278.2004.00828.x.

[11] B.J. Knaus and N.J. Grünwald. VCFR: a package to manipulate and visualize variant call format data in R. *Molecular Ecology Resources*, 17:44–53, 2017. doi: 10.1111/1755-0998.12549.

[12] T. Wei and V. Simko. R package 'corrplot': Visualization of a Correlation Matrix. *GitHub*, 2021. URL `github.com/taiyun/corrplot`. (Version 0.92).

[13] B. Gel and E. Serra. karyoploteR: an R/Bioconductor package to plot customizable genomes displaying arbitrary data. *Bioinformatics*, 33:3088–3090, 2017. doi: 10.1093/bioinformatics/btx346.

[14] B. Gel, A. Diez-Villanueva, E. Serra, M. Buschbeck, M.A. Peinado, and R. Malinverni. regioneR: an R/Bioconductor package for the association analysis of genomic regions based on permutation tests. *Bioinformatics*, 33:3088–3090, 2017. doi: 10.1093/bioinformatics/btx346.

[15] E. Garrison, Z.N. Kronenberg, E.T. Dawson, B.S. Pedersen, and P. Prins. Vcflib and tools for processing the VCF variant call format. *bioRxiv*, 2021.05.21.445151, 2021. doi: 10.1101/2021.05.21.445151.

[16] S. Neph, M.S. Kuehn, A.P. Reynolds, E. Haugen, R.E. Thurman, A.K. Johnson, E. Rynes, M.T. Maurano, J. Vierstra, S. Thomas, R. Sandstrom, R. Humbert, and J.A. Stamatoyannopoulos. BEDOPS: high-performance genomic feature operations. *Bioinformatics*, 28(14):1919–1920, 2012. doi: 10.1093/bioinformatics/bts277.

[17] D. Schaumont, E. Veeckman, F. Van der Jeugt, A. Haegeman, S. van Glabeke, Y. Bawin, L. Lukasiewicz, S. Blugeon, P. Barre, M. de la O Leyva-Perez, S.L. Byrne, P. Dawyndt, and T. Ruttink. Stack Mapping Anchor Points (SMAP): a versatile suite of tools for read-backed haplotyping. *bioRxiv*, 2022.03.10.483555, 2022. doi: 10.1101/2022.03.10.483555.

[18] D.W. Barnett, E.K. Garrison, A.R. Quinlan, M.P. Strömberg, and G.T. Marth. BamTools. *Bioinformatics*, 27(12):1691–1692, 2011. doi: 10.1093/bioinformatics/btr174.

[19] F. Ronquist, M. Teslenko, P. van der Mark, D.L. Ayres, A. Darling, S. Höhna, B. Larget, L. Liu, M.A. Suchard, and J.P. Huelsenbeck. MRBAYES 3.2: Efficient Bayesian phylogenetic inference and model selection across a large model space. *Systemic Biology*, 61:539–542, 2012. doi: 10.1093/sysbio/sys029.

[20] A. Rambaut, A.J. Drummond, D. Xie, G. Baele, and M.A. Suchard. Posterior Summarization in Bayesian Phylogenetics Using Tracer 1.7. *Systemic Biology*, 67:901–904, 2018. doi: 10.1093/sysbio/syy032.

[21] A.M. Kozlov, D. Darriba, T. Flouri, B. Morel, and A. Stamatakis. RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. *Bioinformatics*, 35:4453–4455, 2019. doi: 10.1093/bioinformatics/btz305.

[22] L. Nguyen, H.A. Schmidt, A. von Haeseler, and B.Q. Minh. IQ-TREE: A Fast and Effective Stochastic Algorithm for Estimating Maximum-Likelihood Phylogenies. *Molecular Biology and Evolution*, 32:268–274, 2014. doi: 10.1093/molbev/msu300.

[23] C. Zhang, M. Rabiee, E. Sayyari, and S. Mirarab. ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics*, 19:153, 2018. doi: 10.1186/s12859-018-2129-y.

[24] S. Mirarab. Species Tree Estimation Using ASTRAL: Practical Considerations. *arXiv*, arXiv(q-bio.PE):1904.03826v2, 2019. doi: 10.48550/arXiv.1904.03826.

[25] M. Rabiee, E. Sayyari, and S. Mirarab. Multi-Allele Species Reconstruction Using ASTRAL. *Molecular Phylogenetics and Evolution*, 130:286–296, 2019. doi: 10.1016/j.ympev.2018.10.033.

[26] E. Andersen. VCF-kit: Assorted utilities for the variant call format. *[Online]*, 2020. URL `github.com/AndersenLab/VCF-kit`.

[27] X. Zheng, D. Levine, J. Shen, S. Gogarten, C. Laurie, and B. Weir. A High-performance Computing Toolset for Relatedness and Principal Component Analysis of SNP Data. *Bioinformatics*, 28:3326–3328, 2012. doi: 10.1093/bioinformatics/bts606.

[28] A. Rambaut. FigTree. *[Online]*, 2021. URL `github.com/rambaut/figtree`.

[29] D. Kahle and H. Wickham. ggmap: Spatial Visualization with ggplot2. *The R Journal*, 5:144–161, 2013. doi: 10.32614/RJ-2013-014.

[30] D.E. Wood and S.L. Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, 15, 2014. doi: 10.1186/gb-2014-15-3-r46.

[31] D.E. Wood, J. Lu, and B. Langmead. Improved metagenomic analysis with Kraken 2. *Genome Biology*, 20, 2019. doi: 10.1186/s13059-019-1891-0.

[32] J. Lu. KrakenTools. *GitHub*, 2022. URL `github.com/jenniferlu717/KrakenTools`.

[33] D. Kim, L. Song, F.P. Breitwieser, and S.L. Salzberg. Centrifuge: rapid and sensitive classification of metagenomic sequences. *Genome Research*, 26:1721–1729, 2016. doi: 10.1101/gr.210641.116.

[34] J.M. Martí. Recentrifuge: Robust comparative analysis and contamination removal for meta genomics. *PLOS Computational Biology*, 15:e1006967, 2019. doi: 10.1371/journal.pcbi.1006967.

[35] B.D. Ondov, N.H. Bergman, and A.M. Phillippy. Interactive metagenomic visualisation in a Web browser. *BMC Bioinformatics*, 12:385, 2011. doi: 10.1186/1471-2105-12-385.

[36] Q. Zhu, U. Mai, W. Pfeiffer, S. Janssen, F. Asnicar, J.G. Sanders, P. Belda-Ferre, G.A. Al-Ghalith, E. Kopylova, D. McDonald, T. Kosciolek, J.B. Yin, S. Huang, N. Salam, J. Jiao, Z. Wu, Z.Z. Xu, K. Cantrell, Y. Yang, E. Sayyari, M. Rabiee, J.T. Morton, S. Podell, D. Knights, W. Li, C. Huttenhower, N. Segata, L. Smarr, S. Mirarab, and R. Knight. Phylogenomics of 10,575 genomes reveals evolutionary proximity between domains Bacteria and Archaea. *Nature Communications*, 10:5477, 2019. doi: 10.1038/s41467-019-13443-4.

[37] J.N. Paulson, N.D. Olson, D.J. Braccia, J. Wagner, H. Talukder, M. Pop, and H.C. Bravo. metagenomeSeq: Statistical analysis for sparse high-throughput sequencing. *Bioconductor Package*, [Online] Available at, 2013. URL `https://cbcb.umd.edu/software/metagenomeSeq`.

[38] J.N. Paulson, O.C. Stine, H.C. Bravo, and M. Pop. Differential abundance analysis for microbial marker-gene surveys. *Nature Methods*, 10:1200–1202, 2013. doi: 10.1038/nmeth.2658.

[39] Z. Zhang. KaKs_calculator 3.0: Calculating selective pressure on coding and non-coding sequences. *Genomics, Proteomics Bioinformatics*, 2022. ISSN 1672-0229. doi: 10.1016/j.gpb.2021.12.002.

[40] K. Wang, M. Li, and H. Hakonarson. ANNOVAR: Functional annotation of genetic variants from next-generation sequencing data. *Nucleic Acids Research*, 38:e164, 2010. doi: 10.1093/nar/gkq603.

[41] T.D. Wu and C.K. Watanabe. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics*, 21:1859–1875, 2005. doi: 10.1093/bioinformatics/bti310.

[42] H. Yang and K. Wang. Genomic variant annotation and prioritization with ANNOVAR and wANNOVAR. *Nature Protocols*, 10:1556–1566, 2015. doi: 10.1038/nprot.2015.105.

[43] W. McLaren, L. Gil, S.E. Hunt, H.S. Riat, G.R. Ritchie, A. Thormann, P. Flicek, and F. Cunningham. The Ensembl Variant Effect Predictor. *Genome Biology*, 17:122, 2016. doi: 10.1186/s13059-016-0974-4.