DArTseq_pipeline

Sander de Backer
Meise Botanic Garden

2022

Contents

1 Diversity Arrays Technology Pipeline **2**
- 1.1 DArT Sequencing . . . 2
- 1.2 DArTseq Analysis Pipeline . . . 3
  - 1.2.1 Ubuntu for Windows . . . 4
  - 1.2.2 Dependencies and Tools . . . 4
  - 1.2.3 Preparation and Scripts . . . 6
  - 1.2.4 DArTseq Analysis . . . 7

2 Variant Call Format **8**
- 2.1 VCF Statistics . . . 8
- 2.2 VCF Data Visualisation . . . 9
  - 2.2.1 Sample-Variant Matrix . . . 9
  - 2.2.2 Variant Correlations . . . 10
  - 2.2.3 Variant Position . . . 11

3 Phylogenetics **13**
- 3.1 The Missing Data Problem . . . 13
- 3.2 Neutral Genetic Markers . . . 15
- 3.3 SNP Filtering . . . 15
- 3.4 Quick Phylogenetics . . . 15
  - 3.4.1 NEWICK Tree Format . . . 15
  - 3.4.2 Tree Visualisation . . . 17
- 3.5 Bayesian Inference of Phylogeny . . . 18
- 3.6 Randomised Axelerated Maximum Likelihood (WorkInProgress) 23

4 Metagenomics **24**
- 4.1 Classification . . . 24
  - 4.1.1 Kraken2 . . . 24
  - 4.1.2 Centrifuge . . . 26
  - 4.1.3 Recentrifuge . . . 28
- 4.2 Visualisation . . . 28
  - 4.2.1 Kronatools . . . 29
  - 4.2.2 WebOfLife & metagenomeSeq . . . 29
  - 4.2.3 Pavian (WorkInProgress) . . . 30
- 4.3 De Novo Assemblies . . . 30
- 4.4 The Endogenous Banana Streak Virus (WorkInProgress) . . . 30

5 BEAUti and the BEAST **32**
- 5.1 BEAUti XML File . . . 32
- 5.2 BEAST2 Analysis . . . 33
- 5.3 LogCombiner . . . 33
- 5.4 Tracer . . . 33
- 5.5 TreeAnnotator . . . 33
- 5.6 FigTree . . . 33

6 **Functional Proteomics** **34**

7 **fastStructure** **35**

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). 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**).

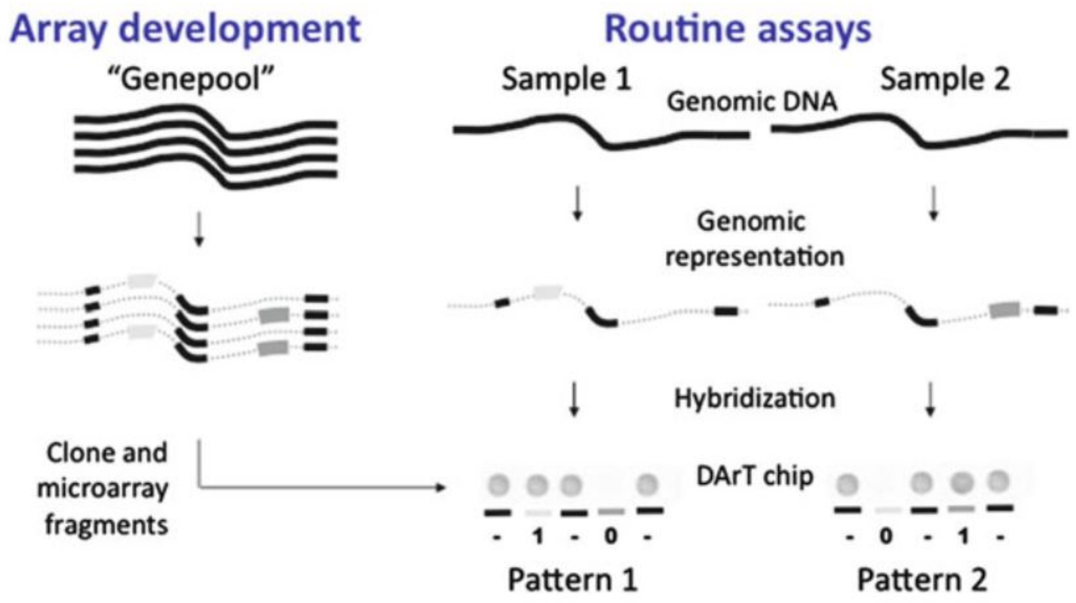
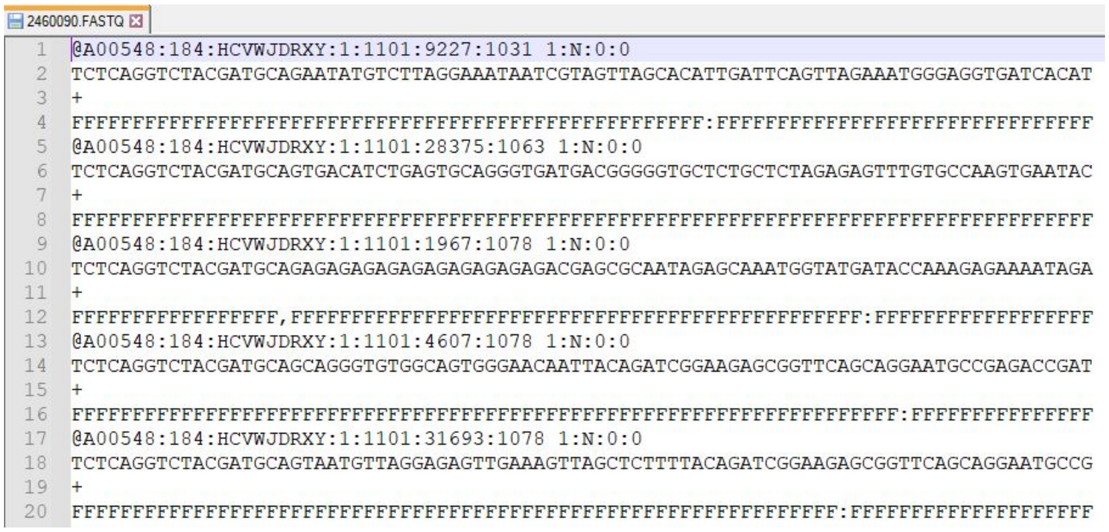


Figure 1: Principles of a DArT analysis. Genomic complexity reduction with restriction enzymes, and hybridisation between genepool and samples on the DArT array.

DArTseq is an effort in "genotyping by sequencing" the complexity-reduced fractions of plant genomes. 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.

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 is your friend!

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*, *pip*, or *conda*. The *python3-pip* package enables installation with pip. The installation of *conda* is discussed further on.

```
# command to return to home directory (from wherever)
# cd = Change Directory
cd
# installations with sudo
sudo apt install python3 python3-pip
sudo apt install fastqc
sudo apt install bwa
sudo apt install vcftools
# 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, 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.

```
# 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 the package
./configure --prefix=/home/USER/htslib-1.x
make
make install
```

For *conda* installations, go to https://docs.conda.io/projects/conda/en/latest/user-guide/index.html and follow the regular installation instructions for Linux to install Miniconda3. When this is correctly installed, use the commands below for conda installations.

```
# update conda packages
cd
conda update --all
# only for CondaHTTPError, OTHERWISE SKIP!
# change permissions of your conda-installation
chmod -R 777 ~/miniconda3
# update conda itself
conda update -n base conda -c anaconda
# install FreeBayes with conda
conda install -c bioconda freebayes # check the GitHub page for latest version
# check version
freebayes
```

Depending on the version of conda, and compatibility with the computing system, the latest version of FreeBayes may not be available. Running the basic *freebayes* command also shows the version, and should be the one corresponding to the latest release on the GitHub repository (https://github.com/freebayes/freebayes). If this is not the case, the latest static version for Linux can be downloaded from GitHub and simply be extracted to replace the installed (older) version. To find the location of installation, use the *which* command.

```
# only if installed version =/= latest GitHub version, OTHERWISE SKIP!
# move the downloaded file into the Ubuntu home directory and unpack
cd
mv /mnt/c/Users/USER/Downloads/freebayes-1.3.6-linux-amd64-static.gz .
gunzip freebayes-1.3.6-linux-amd64-static.gz
# find the FreeBayes installation folder
which freebayes
```

```
# replace FreeBayes with pre-built static
mv freebayes-1.3.6-linux-amd64-static /home/USER/miniconda3/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 looking into the .bashrc file in the home directory of the Ubuntu terminal. Remember to fill in your username!

```
# from home directory
nano .bashrc
# from anywhere else
nano $HOME/.bashrc
# standard example of $PATH in .bashrc
export PATH="$PATH:$HOME/bin:/usr/bin"
# with path of GBprocesS added
export PATH="$PATH:$HOME/bin:/usr/bin:/home/USER/miniconda3/bin"
```

Picard: go to https://broadinstitute.github.io/picard/ and download the 'Latest Release' (file is always named 'picard.jar' regardless of the version). Place this .jar file in a directory easily accessible (e.g. subdirectory called 'Dependencies' in the directory of the raw FASTQ files).

GATK4: go to https://gatk.broadinstitute.org/hc/en-us and 'Download the latest version of GATK' (GATK4.2). 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 place it in a directory easily accessible (e.g. subdirectory called 'Dependencies' in the directory of the raw FASTQ files).

1.2.3 Preparation and Scripts

The DArTseq_pipeline will process FASTQ.gz files into Variant Call Format (VCF) files. For the pipeline to run smoothly, follow this setup. Begin with placing the dependencies mentioned above (picard and GATK) in a folder called 'Dependencies'. Besides the dependencies, 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 (in the 'Dependencies' folder).

```
# 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
```

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. The Identify_barcodes Python script reads all fastq files (gzipped or not) in the directory, writing a TableOfBarcodes.fasta file that contains the barcode of every sample in the directory. 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.

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/.

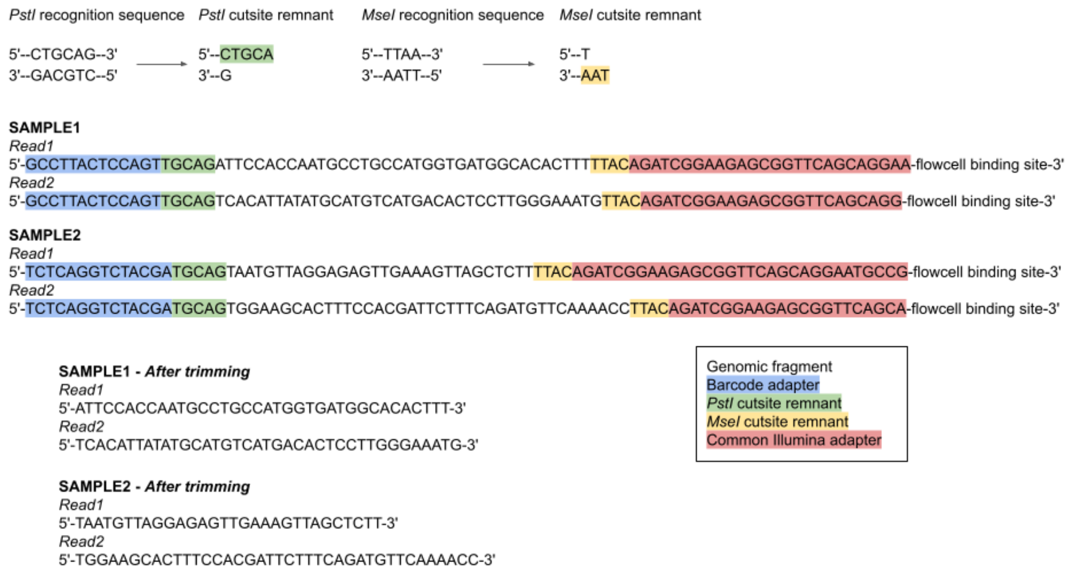

PstI recognition sequence
PstI cutsite remnant
MseI recognition sequence
MseI cutsite remnant
5'--CTGCAG--3'
3'--GACGTC--5'
5'--CTGCA
3'--G
5'--TTAA--3'
3'--AATT--5'
5'--T
3'--AAT
SAMPLE1
Read1
5'-GCCTTACTCCAGTTGCAGATTCCACCAATGCCTGCCATGGTGATGGCACACTTTTTACAGATCGGAAGAGCGGTTCAGCAGGAA-flowcell binding site-3'
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.

1.2.4 DArTseq Analysis

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

Comments

An alternative pipeline can be found at https://github.com/CathyBreton/Genomic_Evolution. 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/3M0oq8z.

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.

References

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

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

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

Risterucci, A., Hippolyte, I., Perrier, X., Xia, L., Caig, V., Evers, M., Huttner, E., Kilian, A., Glaszmann, J. **(2009)** Development and assessment of Diversity Arrays Technology for high-throughput DNA analyses in Musa. *Theoretical and Applied Genetics.* **119**(6): 1093–1103. doi.org/10.1007/s00122-009-1111-5

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

2 Variant Call Format

The Variant Call Format (VCF) file stores genetic sequence variation in text file format. In this case 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. More information available at https://github.com/sanderdebacker/DArTseq_pipeline (VariantCallFormat.pdf).

2.1 VCF Statistics

Many options are available for a large variety of statistics. Mainly VCFtools and BCFtools 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 dataset. For this, the single sample VCF files need to be combined into a multi-sample VCF file. On how to merge VCF files, look under the section Phylogenetics where it is explained in detail. For each tool, the main statistical output options are explained as also available on their respectful manual pages.

VCFtools (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

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. This also allocates a colour code to the sample-variant combination, highlighting the quality and probability of the mutation. 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 (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

```r
install.packages('vcfR') #only once
library(vcfR)
vcf <- read.vcfR("VCF.vcf.gz", verbose = TRUE ) #load VCF file
dna <- ape::read.dna("REF.fasta", format = "fasta") #reference fasta file
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)
head(chrom)
```

```
dp <- extract.gt(chrom, element="DP", as.numeric=TRUE)
rownames(dp) <- 1:nrow(dp) #number the variants (not required)
head(dp)
heatmap.bp(dp) #all variants
#generate PNG image of heatmap
png(filename="VCF_SNPs.png", height=5000, width=5000, res=300)
heatmap.bp(dp[1001:1500,]) #only variant 1001 to 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 one are plotted against the samples in the dataset.

```
dna <- ape::read.dna("REF.fasta", format = "fasta") #reference fasta file
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.packages("tidyverse") #only once
library(tidyverse)
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
matrix <- data.matrix(vcf)
matrix <- t(matrix)
cor <- cor(matrix)
col <- colorRampPalette(c("blue", "white", "red"))(20)
#generate PNG image of correlation 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. 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.

```r
install.packages("corrplot") #only once
library(corrplot)
corrplot(cor, type = "upper", tl.col = "black", tl.srt = 45)
```

2.2.3 Variant Position

The requirements for a visualisation with KaryoploteR 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 Gene-based 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. Here, plotting as points, density, and only those related to a gene of interest are shown.

```r
install.packages("BiocManager") #only once
BiocManager::install("karyoploteR") #only once
library(karyoploteR)
vars <- read.table("VCF.vcf") #read VCF
custom.genome <- toGRanges("ChromStructure.txt") #construct genome
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)
#plotting genomic regions
regions.reads <- toGRanges("GENE.bed")
kp <- plotKaryotype(genome=custom.genome)
kpPlotRegions(kp, data=regions.reads)
#plotting 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*")
```

```r
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.

```r
BiocManager::install("regioneR") #only once
library(regioneR)
library(karyoploteR)
set.seed(123456)
vcf <- read.table("VCF.vcf")
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$pval)
snps.ranges <- toGRanges(snps)
custom.genome <- toGRanges("ChromStructure.txt")
kp <- plotKaryotype(custom.genome, plot.type=4)
transf.pval <- -log10(snps.ranges$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)
```

References

Danecek, P., Auton, A., Abecasis, G., Albers, C.A., Banks, E., DePristo, M.A., Handsaker, R., Lunter, G., Marth, G., Sherry, S.T., McVean, G., Durbin, R., 1000 Genomes Project Analysis Group **(2011)** The Variant Call Format and VCFtools. *Bioinformatics.* **27**(15): 2156–2158. doi.org/10.1093/bioinformatics/btr330

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

Gel, B., Serra, E. **(2017)** karyoploteR: an R/Bioconductor package to plot customizable genomes displaying arbitrary data. *Bioinformatics.* **33**(19): 3088-3090. doi.org/10.1093/bioinformatics/btx346

Knaus, B.J., Grünwald, N.J. **(2017)** VCFR: a package to manipulate and visualize variant call format data in R. *Molecular Ecology Resources.* **17**(1): 44-53. doi.org/10.1111/1755-0998.12549

Wei, T., Simko, V. **(2021)** R package 'corrplot': Visualization of a Correlation Matrix (Version 0.92). Available from https://github.com/taiyun/corrplot

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 and failing other criteria. While bcftools and vcftools take VCFs as direct input, useful in adapting and modifying VCF files, the libvcflib packages are necessary for Python scripts to parse and manipulate VCFs.

Ubuntu terminal

```
# install packages
sudo apt install bcftools
sudo apt install vcftools
sudo apt-get install libvcflib-tools libvcflib-dev
# for OSX
brew install bcftools
brew install vcftools
brew install 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 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.

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 'NA/NaN/Inf' errors in the subsequent analyses.

Bash script

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

3.1 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 reads vary between samples. To prevent wrongful interpretation of missing reads/SNPs in samples analysed with lower depth in comparison to samples analysed with higher depth, the final VCF can be filtered based on reads present in 90-99% of the samples (depending on the allowed percentage of missing data). The SMAP delineate tool compares all BAM files (in BAM/ 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.

```
# 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 ‘out of memory’ error, the amount of BAM files it tries to process requires an amount of memory which is too large for the processing system. The BAM files of the samples can be split into chromosomes with the *bamtools split* function. 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 indexing the multi-sample VCF with *tabix*, it can be filtered with the merged BED file using the *bcftools filter* function.

```
#!/bin/bash
# remove unmapped reads
for file in ./*bam; do
        samtools view $file chr{01..11} -b > "${file%.bam}.chr.bam"
done
# split BAMs into chromosomes
for file in ./*chr.bam; do
        bamtools split -in $file -reference
done
# clean up directory
for i in {01..11}; do
        mkdir "chrom$i"
        mv *chr$i* ./"chrom$i"/
done
# index all the new BAMs
for file in ./chrom*/*bam; do
        samtools index $file
done
# smap delineate for each chromosome
for i in {01..11}; do
        smap delineate "chrom$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 ReadPositionsChr* > CompletePositions.bed
```

```
# index VCF
tabix -f -h VCF.vcf.gz
# filter VCF with bcftools
bcftools filter -R CompletePositions.bed -Oz VCF.vcf.gz > VCF_filtered.vcf.gz
```

3.2 Neutral Genetic Markers

Assessments of population genetics and historical demographics have traditionally been based on neutral markers while explicitly excluding adaptive markers. Therefore, the difference between exonic and intronic SNPs has to be made. 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 in those ranges.

Annotation files are often in *gff3* (General Feature Format 3) format. With the BEDOPS utilities, this can easily be converted to BED format, which can be used by the *bcftools filter* function to select SNPs. Follow the installation on https://bedops.readthedocs.io/en/latest/content/installation.html#linux.

```
# BEDOPS command
gff2bed --input=gff --output=bed < GFF3.gff3 > GFF3.bed
# index and filter VCF with bcftools
tabix -f -h VCF.vcf.gz
bcftools filter -R GFF3.bed -Oz VCF.vcf.gz > VCF_filtered.vcf.gz
```

3.3 SNP Filtering

The filtration steps are a combination of GATK4 SelectVariants and a custom FilterVCF.py script, by Yves Bawin from the gitlab repository (Scripts folder) gitlab.com/ybawin/sequence-data-processing-tetraploids. It is also available for download at github.com/sanderdebacker/DArTseq_pipeline. Annotation of the genotype likelihood (GL) tag is required for GATK SelectVariants. The first GATK step excludes the non-variant calls and the second GATK step removes the multi-allelic SNPs (more than two alleles are probably read errors). The FilterVCF script removes variants based on allele depth (-a), the completeness across all samples (-c), the total read depth across all samples (-d), and the genotype quality (-q).

```
#annotate genotype likelihood tag to prevent errors in next step
bcftools annotate -x 'FORMAT/GL' VCF_filtered.vcf > VCF_GL.vcf
#remove 0/0 non-variant calls
gatk SelectVariants -R REF.fasta -V VCF_GL.vcf --exclude-non-variants -O VCF_GL_ENV.vc
#remove multi-allelic variants (most likely read errors)
gatk SelectVariants -R REF.fasta -V VCF_GL_ENV.vcf --restrict-alleles-to BIALLELIC -O
#filter on allele depth, completeness, total read depth, genotype quality
python3 FilterVCF.py --vcf VCF_GL_ENV_RA.vcf -a 3 -c 0.95 -d 10 -q 20
```

3.4 Quick Phylogenetics

Widely used phylogenetic tools like MrBayes and RAxML have long processing times. First, we discuss some ways to generate a cladogram fast to give a quick impression.

3.4.1 NEWICK Tree Format

This file is a mathematical representation of graph-theoretical trees with edge lengths using parentheses and commas. It can be visualised in multiple ways.

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. 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 to produce a NJ (neighbour-joining) of UPGMA (unweighted pair group method with arithmetic-mean) tree, with output in NEWICK format.

```
# install numpy & vcf-kit
pip install numpy
pip install VCF-kit
# install homebrew for linux (don't forget to add to PATH)
/bin/bash -c "$(curl -fsSL \
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
# install dependencies
vk setup
# phylogenetic tree generation
vk phylo tree [nj|upgma] VCF.vcf > VCF.newick
```

SNPRelate The R package *SNPRelate* provides a binary format for single-nucleotide polymorphism (SNP) data utilizing CoreArray Genomic Data Structure (GDS) data files. Together with packages *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("SNPRelate")) # only once
BiocManager::install(c("gdsfmt")) # 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
```

ASTRAL-III For multiple gene trees, ASTRAL-III can be used. 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, i.e., 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.

```
# astral command
java -jar astral.jar -i VCF.newick -o VCF.tre -a species.txt -t 2 2>astral.log
```

3.4.2 Tree Visualisation

FigTree Download and install FigTree from its GitHub repository: https://github.com/rambaut/figtree. 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. Other tree visualisation options 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, 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))
```

3.5 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. Bayesian inference is 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. 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
# open .bashrc file to add installation to $PATH
nano $HOME/.bashrc
```

First, the merged VCF file needs to be converted into NEXUS format. For this, download the vcf2phylip Python script from its GitHub repository (command below). The script will be in /home/USER/vcf2phylip/ when following the instructions. Add the installation to the $PATH for convenience. For binary NEXUS files, use the -b argument instead of -n. To disable the standard PHYLIP output, use the -p argument.

```
# download script
cd
git clone https://github.com/edgardomortiz/vcf2phylip
# open .bashrc file to add installation to $PATH
nano $HOME/.bashrc
# vcf2phylip command
python vcf2phylip.py -i VCF.vcf --output-folder . --output-prefix PREFIX -n -p
```

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 + + (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.*

```
# info about commands
> 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 *help mcmc* 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.

```
# info about command
> help mcmc
# run analysis with adjusted values
> 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**). 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.

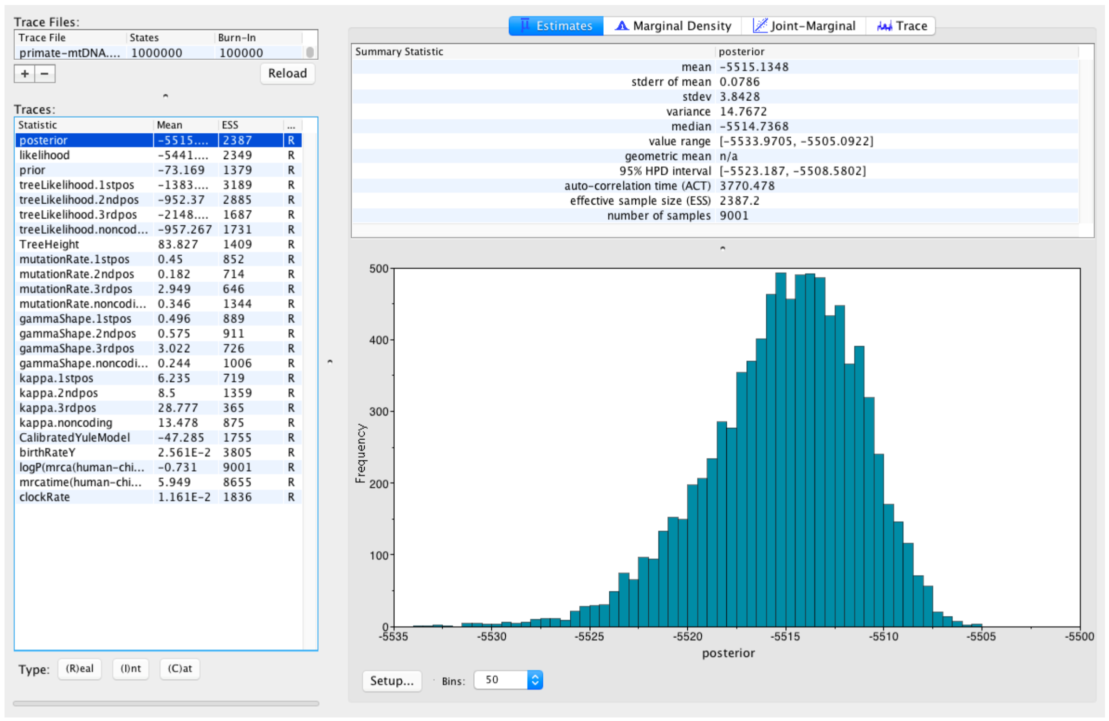
```
# summarise sampled parameters values
> sump
```

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
```

Burnin 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.

Generations 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



Trace Files:

Trace File	States	Burn-In
primate-mtDNA....	1000000	100000

Traces:

Statistic	Mean	ESS	...
posterior	-5515....	2387	R
likelihood	-5441....	2349	R
prior	-73.169	1379	R
treeLikelihood.1stpos	-1383....	3189	R
treeLikelihood.2ndpos	-952.37	2885	R
treeLikelihood.3rdpos	-2148....	1687	R
treeLikelihood.noncod...	-957.267	1731	R
TreeHeight	83.827	1409	R
mutationRate.1stpos	0.45	852	R
mutationRate.2ndpos	0.182	714	R
mutationRate.3rdpos	2.949	646	R
mutationRate.noncodi...	0.346	1344	R
gammaShape.1stpos	0.496	889	R
gammaShape.2ndpos	0.575	911	R
gammaShape.3rdpos	3.022	726	R
gammaShape.noncodi...	0.244	1006	R
kappa.1stpos	6.235	719	R
kappa.2ndpos	8.5	1359	R
kappa.3rdpos	28.777	365	R
kappa.noncoding	13.478	875	R
CalibratedYuleModel	-47.285	1755	R
birthRateY	2.561E-2	3805	R
logP(mrca(human-chi...	-0.731	9001	R
mrcatime(human-chi...	5.949	8655	R
clockRate	1.161E-2	1836	R

Summary Statistic	posterior
mean	-5515.1348
stderr of mean	0.0786
stdev	3.8428
variance	14.7672
median	-5514.7368
value range	[-5533.9705, -5505.0922]
geometric mean	n/a
95% HPD interval	[-5523.187, -5508.5802]
auto-correlation time (ACT)	3770.478
effective sample size (ESS)	2387.2
number of samples	9001

Figure 4: Example of the Tracer interface when analysing the log files of a BEAST analysis, similar to the MrBayes run1.p and run2.p log files.

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.

Covarion This forces the use of a covarion-like model of substitution for nucleotide or amino acid data. The valid options are "yes" and "no". The covarion model allows the rate at a site to change over its evolutionary history. Specifically, the site is either on or off. When it is off, no substitutions are possible. When the process is on, substitutions occur according to a specified substitution model (specified using the other *lset* options). The covarion hypothesis of molecular evolution proposes that selective pressures on an amino acid or nucleotide site change through time, thus causing changes of evolutionary rate along the edges of a phylogenetic tree.

Nucmodel This 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).

Substitution Type *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.

Tree Age Prior This parameter (*treeagepr*) 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: *prset treeagepr = 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.

Clock Rate Prior This parameter (*clockratepr*) 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 *prset clockratepr = 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 *prset clockratepr = 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, *prset clockratepr = 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.

3.6 Randomised Axelerated Maximum Likelihood (WorkInProgress)

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) 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. In RAxML-NG, statistical support is obtained from separate bootstrap searches. In comparison to previous versions, RAxML-NG generally runs faster.

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.

```
# 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. Convert the multi-sample VCF file to PHYLIP format with the vcf2phylip Python script. The optional input tree format is NEWICK, and must not always be comprehensive, i.e., need not contain all taxa of the alignment.

```
# vcf2phylip conversion
python3 vcf2phylip.py -i VCF.vcf --output-folder . --output-prefix PREFIX
```

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

References

Andersen, E. **(2020)** VCF-kit: Assorted utilities for the variant call format [Online]. Available at github.com/AndersenLab/VC
kit

Mirarab, S. **(2019)** Species Tree Estimation Using ASTRAL: Practical Considerations. *arXiv*:1904.03826v2 (q-bio.PE)

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

Roch, S., Nute, M., Warnow, T. **(2019)** Long-Branch Attraction in Species Tree Estimation: Inconsistency of Partitioned Likelihood and Topology-Based Summary Methods. *Systematic Biology.* **68**(2): 281–297. doi.org/10.1093/sysbio/syy061

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

Schliep, K.P. **(2011)** phangorn: phylogenetic analysis in R. *Bioinformatics.* **27**(4): 592-593. [Online] Available at https://github.com/KlausVigo/phangorn

Van der Auwera, G.A., Carneiro, M., Hartl, C., Poplin, R., del Angel, G., Levy-Moonshine, A., Jordan, T., Shakir, K., Roazen, D., Thibault, J., Banks, E., Garimella, K., Altshuler, D., Gabriel, S., DePristo, M. **(2013)** 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. doi.org/10.1002/0471250953.bi1110s43

Zhang, C., Rabiee, M., Sayyari, E., Mirarab, S. **(2018)** ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics.* **19**(S6): 153. doi.org/10.1186/s12859-018-2129-y

Zheng, X., Levine, D., Shen, J., Gogarten, S., Laurie, C., Weir, B. **(2012)** A High-performance Computing Toolset for Relatedness and Principal Component Analysis of SNP Data. *Bioinformatics.* **28**(24): 3326-3328. doi.org/10.1093/bioinformatics/bts606

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.

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

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 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

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). The graph below shows the classification richness in function of the confidence.

Figure: 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.

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.

```bash
#!/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
```

Comments

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. 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.

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. 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
git clone https://github.com/DaehwanKimLab/centrifuge
cd ./centrifuge/
sudo make install prefix=$HOME/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]
```

```
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.

```
#!/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

Same first point as Kraken2 (see above). Although Centrifuge databases take much less disk space, they have the downside that building databases takes much more time (see example above). 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/centrifuge/issues/221

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. 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, 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) 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.

```
#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
biom convert -i out.all.tsv -o out.all.biom --table-type="OTU table" --to-hdf5
```

```r
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 paired-end reads is present (e.g. unstacked paired-end data of shotgun sequencing). DArTseq generates stacked 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)
```

References

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

Martí, J.M. **(2019)** Recentrifuge: Robust comparative analysis and contamination removal for meta genomics. *PLOS Computational Biology.* **15**(4): e1006967. doi.org/10.1371/journal.pcbi.1006967

Ondov, B.D., Bergman, N.H., Phillippy, A.M. **(2011)** Interactive metagenomic visualisation in a Web browser. *BMC Bioinformatics.* **12**(1):385. doi.org/10.1186/1471-2105-12-385

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

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

Wood, D.E., Salzberg, S.L. **(2014)** Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology.* **15**(R46) doi.org/10.1186/gb-2014-15-3-r46

Wood, D.E., Lu, J., Langmead, B. **(2019)** Improved metagenomic analysis with Kraken 2. *Genome Biology.* **20**(257) doi.org/10.1186/s13059-019-1891-0

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

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

For duplication and transpositions: Plant Genome Duplication Database at http://pdgd.njau.edu.cn:8080/ Cysteine3-Histidine Zinc-Finger (CCCH-ZF) Gene Family Mazumdar, P., Lau, S., Wee, W., Singh, P., Harikrishna, J. (2017) Genome-wide Analysis of the CCCH Zinc-Finger Gene Family in Banana (Musa acuminata): An Insight Into Motif and Gene Structure Arrangement, Evolution and Salt Stress Responses. Tropical Plant Biology. 10: 4 doi.org/10.1007/s12042-017-9196-5 For sequences (pep and nt): http://planttfdb.gao-lab.org/family.php?sp=Macfam=C3H Drought Tolerance Related Genes Van Wesemael, J., Hueber, Y., Kissel, E., Campos, N., Swennen, R., Carpentier, S. (2018) Homeolog expression analysis in an allotriploid non-model crop via integration of transcriptomics and proteomics. Scientific Reports. 8(1): 1353. doi.org/10.1038/s41598-018-19684-5 Ethylene Response Factors (ERFs), Ethylene Insensitive-Like (EIL) Jourda, C., Cardi, C., Mbeguie-A-Mbeguie, D., Bocs, S., Garsmeur, O., D'Hont, A., Yahiaoui, N. (2014) Expansion of banana (Musa acuminata) gene families involved in ethylene biosynthesis and signalling after lineage-specific whole-genome duplications. New Phytologist. 202: 986–1000. doi.org/10.1111/nph.12710 For sequences (pep and nt): http://planttfdb.gao-lab.org/family.php?fam=EIL For sequences (pep and nt): http://planttfdb.gao-lab.org/family.php?sp=Macfam=ERF MADS-box family Elitzur, T., Vrebalov, J., Giovannoni, J.J., Goldschmidt, E.E., Friedman, H. (2010) The regulation of MADS-box gene expression during ripening of banana and their regulatory interaction with ethylene. Journal of Experimental Botany. 61(5): 1523-35. doi.org/10.1093/jxb/erq017. Liu, J., Xu, B., Hu, L., Li, M., Su, W., Wu, J., Yang, J., Jin, Z. (2009) Involvement of a banana MADS-box transcription factor gene in ethylene-induced fruit ripening. Plant Cell Reports. 28(1): 103–111. doi.org/10.1007/s00299-008-0613-y For sequences (pep and nt): http://planttfdb.gao-lab.org/family.php?sp=Macfam=M-type$_M$*ADSMYBTranscriptionFactors : Pucker, B., Pandey, A., Weisshaar, B., Strack*
MYBgenefamilyinbanana(Musaacuminata) : Genome−wideidentification, classificationandexpressionpatterns.PLoSO
e0239275.doi.org/10.1371/journal.pone.0239275Tan, L., Ijaz, U., Salih, H., Cheng, Z., NiWinHtet, N., Ge, Y., Azeem, F.(202
WideIdentificationandComparativeAnalysisofMYBTranscriptionFactorFamilyinMusaacuminataandMusabalbisiana.
413.doi.org/10.3390/plants9040413Forsequences(pepandnt) : http : //planttfdb.gao−lab.org/family.php?sp =
Macfam = MYBTCPTranscriptionsFactors : SnchezMoreano, J.P., Xu, X., AucapiaCriollo, C.B., Chen, X., Lin, Y., Mu
WideIdentificationandComprehensiveAnalysesofTCPGeneFamilyinBanana(MusaL.).TropicalPlantBiology.14(2) :
180˘202.doi.org/10.1007/s12042 − 021 − 09281 − 8Forsequences(pepandnt) : http : //planttfdb.gao −
lab.org/family.php?sp = Macfam = TCPWRKYGeneFamilyGoel, R., Pandey, A., Trivedi, P.K., Asif, M.H.(2016)Genon
WideAnalysisoftheMusaWRKYGeneFamily : EvolutionandDifferentialExpressionduringDevelopmentandStress.Fro
299.doi.org/10.3389/fpls.2016.00299Forsequences(pepandnt) : http : //planttfdb.gao−lab.org/family.php?sp =
Macfam = WRKYDiscoverGeneVariantsFirst, weselectafastafilecontainingtheentiregeneticregionofageneofinterest.!

REF=./Reference/Clathrin.fasta CDS=./Reference/Clathrin$_C$*DS.fastaGENE = Clathrin*

cat *REF*CDS > tempGeneCDS.fasta muscle -in tempGeneCDS.fasta -out tempGeneAlignment.fasta BED=./Reference/Clathrin$_C$*DS.bed*

for file in ./*GBprocesS.fastq.gz; do bwa mem *REF*file > "*filedone*

for file in ./**GENE.sam*; *doSAMPLE* = '*basename*file .sam' java -jar ./Dependencies/picard.jar AddOrReplaceReadGroups -I *file − O*"file done

for file in ./**GENE.bam*; *dofreebayes − f*REF –genotype-qualities –strict-vcf -b *file − v*"file done

for file in ./**GENE.FB.vcf*; *do./Dependencies/GATK4.2/gatkIndexFeatureFile − I*file ./Dependencies/GATK4.2/gatk FastaAlternateReferenceMaker -L *BED − R*REF -V *file − O*"filedone

awk 'print /$^>$/?" > "*FILENAME* :0' **GENE.fasta > samples.fasta*

muscle -in samples.fasta -clwstrict -out samples.aln

/home/sander/kakscalculator2/bin/KaKs$_C$*alculator − isamples.aln − osamples.kaks*

ANNOVAR Download annovar by registering an email address on the annovar website 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 (plantentuinmeise.be should already be implemented in the authorised suffix list). Unpacking the package will result in an 'annovar' folder containing all the required scripts. tar xvfz annovar.latest.tar.gz Use the galaxy server bed-to-gff convertor on usegalaxy.be to convert the BED file to GFF format. Download the GFF file and use to create ANNOVAR database. conda install -c conda-forge mamba mamba install -c conda-forge -c bioconda cgat-apps 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$_{C}alculatortoolisavailableathttps:$
$//ngdc.cncb.ac.cn/biocode/tools/BT000001/releases/3.0.TheKaKsanalysisrequiresadditionaltools.Theinstallationcomma$
in .bashrc file

For the analysis, the sequences must be in an alignment format. Here, the VCF files are transformed into a single CLUSTALW alignment file, which is a direct input for the KaKs$_{C}alculatorcommand.References:$
$Zhang, Z.(2022)KaKs_{c}alculator3.0 : Calculatingselectivepressureoncodingandnon-codingsequences.Genomics, Proteomi$

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
virtualenv --python=$(which python2) /home/USERNAME/python2env/
source /home/USERNAME/python2env/bin/activate #terminate with "deactivate"
# install dependencies and specify version
pip install numpy==1.16.5
pip install cython==0.27.3
pip install scipy==1.2.1
mkdir /home/USERNAME/bin
cd /home/USERNAME/bin
# download and install latest gsl
wget ftp://ftp.gnu.org/gnu/gsl/gsl-2.7.tar.gz
tar -xf gsl-2.7.tar.gz
cd gsl-2.7
./configure --prefix=/home/USERNAME/bin/gsl-2.7
make
make check
make install
# in .bashrc add, or update exports
export LD_LIBRARY_PATH=$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
source /home/USERNAME/python2env/bin/activate
cd /home/USERNAME/bin
mkdir fastStructure
cd fastStructure
sudo apt-get install python-dev
# close and reopen terminal to refresh new global settings
# install fastStructure
source /home/USERNAME/python2env/bin/activate
cd /home/USERNAME/bin/fastStructure
wget --no-check-certificate https://github.com/rajanil/fastStructure/archive/master.ta
tar -xf master.tar.gz
cd fastStructure-master/vars
python setup.py build_ext -f --inplace
cd ..
```

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

```
#!/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