Population structure and hybridization under contemporary and future climates in a heteroploid foundational shrub species (*Artemisia tridentata*)

EBG data processing and analysis pipeline

Lukas P. Grossfurthner

Contents

Introduction

This document provides the processing and analysis pipeline as published in Grossfurthner et al. (2023), with descriptions of each step and annotations for the utilized code. In Grossfurthner et al. (2023), a polyploid genotyping approach (EBG, Blischak et al. 2019) was utilized to enhance mixed ploidy data analysis.

EBG utilizes genotypes generated with GATK HaplotypeCaller (Van der Auwera & Connor, 2018) for both ploidy levels respectively, filters for shared variants between the ploidy levels and estimates genotypes based

on read counts and per locus error rates using a flat prior on genotypes. After finishing the EBG pipeline, genotype matrices for each ploidy level respectively are generated in a plain text file.

This pipeline was initially developed for diploid/tetraploid data sets, but is now generalized to be expandable higher ploidies, where the limitations lie in the downstream processing software (i.e. PolyRelatedness allows ploidyies <8)

This data set consists of diploid and tetraploid samples of three widely acknowledged *A. tridentata* subspecies, which were sampled along five environmental gradients in the western United States. At each transect, four or five plots with 20 samples each were collected.

The labelling scheme in this data set corresponds to: 1. Character - Site (i.e. C = Castle Rocks State Park, ID) 2. Character - Plot (i.e. C2 = Castle Rocks State Park, ID, Plot Nr. 2) 4-5. Character Sample Nr. (C2_13 = Castle Rocks State Park, ID, Plot Nr. 2, Individual Nr. 13) additionally, technical replicates were included. Replicated samples are appended with "_rep" or "_NA".

This pipeline was written under R version 3.3.6 and tested up tp R version 4.3.1 on a aarch64-apple-darwin20 (64-bit) platform, running under: macOS Ventura 13.5.2.

Setup and data preparation

Installing and loading libraries

```
list.of.packages <- c("gplots","gridExtra","reshape2","tidyverse")

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

library(tidyverse)
library(gridExtra)
library(reshape)
library(gplots)
```

Download data set and source code

The data set available at https://github.com/LukeBotanist/Sagebrush contains all required files and code for the analysis below. To facilitate a smooth workflow, download and setting up directory structure are implemented here.

First, the working directory has to be set and the zipped directory needs to be downloaded.

```
setwd("path/to/working/directory")
```

```
main <- "https://github.com/LukeBotanist/Sagebrush/archive/refs/heads/main.zip"

download.file(main, "main.zip")
unzip(zipfile = "main.zip")
```

After downloading, change to `Sagebrush-main` directory

```
setwd("path/to/working/directory/Sagebrush-main")
```

Create directory structure

The data set contains directories `data` and `src`, to allow the results to be stored in output directories, a results forder and respective subfolders can be generated.

```
dirs2create <- c("results/figures", "results/tables")
for (i in 1:length(dirs2create)){
  dir.create(dirs2create[i], recursive = T)
}
```

Load functions from source file

The functions used for data analyses are stored in the `functions.R` file in the `src` directory. If necessary, the functions can be modified or extended if required.

```
source("src/functions.R")
```

Check filestructure

To assess content and data hierarchy, a function similar to the unix `tree` function (available from https://gist.github.com/jennybc/2bf1dbe6eb1f261dfe60) allows a quick visual inspection and provides a better overview than just listing files.

```
twee()
```

```
## -- data
##    |__EBG_data
##       |__dipl_fin-genos.txt
##       |__dipl_indnames
##       |__shared-variants.txt
##       |__tetra_fin-genos.txt
##       |__tetra_indnames
##    |__PolyRelatedness_data
##       |__end.txt
##       |__header.txt
##       |__sagebrush_example.txt
##    |__STRUCTURE_data
##       |__K4_assignments.txt
## -- README.md
## -- results
##    |__figures
##    |__tables
## -- src
##    |__EBG_pipeline.R
##    |__functions.R
```

Importing files

Reading in all required files: 1. Genotype files of diploids and tetraploids, respectively 2. Sample names of the diploids and tetraploids, respectively 3. Shared variants file as extracted by EBG

```
gt2x <- read.table("data/EBG_data/dipl_fin-genos.txt") # read diploid genotype file
gt4x <- read.table("data/EBG_data/tetra_fin-genos.txt") # read tetraploid genotype file
ind2x <- read.table("data/EBG_data/dipl_indnames") # read diploid sample names - those must be in the s
ind4x <- read.table("data/EBG_data/tetra_indnames") # read tetraploid sample names - those must be in t
shared.vars <- read.table("data/EBG_data/shared-variants.txt") # read shared variants file
```

It is important that the sample names are ordered exactly the same as they are in the vcf file which was used as EBG input! Additionally, we need to ensure that the same number of variants was generated for both files, otherwise data sets won't match.

```
ebg.out.info <- data.frame(ploidy=c("2x", "4x"),
           nr.ind=c(dim(gt2x)[1],dim(gt4x)[1]),
           nr.vars=c(dim(gt2x)[2],dim(gt4x)[2]))
print(ebg.out.info)
```

```
##   ploidy nr.ind nr.vars
## 1     2x    211   19277
## 2     4x    227   19277
```

The diploid data set consists of 211 individuals and 19277 variants, the tetraploid data set contains 227 individuals and 19277 variants. As data sets match inspecting and processing data can be proceeded.

Inspect the file format for the diploid data set:

```
print(head(gt2x[1:5]))
```

```
##   V1 V2 V3 V4 V5
## 1  0  0  0  0  0
## 2  1  0  0  0  0
## 3  0  0  0  0  0
## 4  0  0  0  0  0
## 5  0  0  0  0  0
## 6  0  0  0  0  0
```

File format of the tetraploids:

```
print(head(gt4x[1:6]))
```

```
##   V1 V2 V3 V4 V5 V6
## 1  1  0  0  0  0  0
## 2  0  0  0  0  0  0
## 3  1  0  0  0  0  0
## 4  0  0  0  0  0  0
## 5  0  0  1  0  0  3
## 6  0  1  0  0  0  1
```

The genotypes are ordinally encoded, such as **{0,1,2}** for the diploids, where 0 corresponds to the homozygote reference allele, 1 to the heterozygote and 2 to homozygous alternate allele as described in Blischak (2018). Another way to encode these genotypes is to dominance genotypes as characters **(0=AA, 1=AT, 2=TT)** or as numeric values **(0=11, 1=12, 2=22)**. Similarly, tetraploid genotypes are encoded as **{0,1,2,3,4}**, and can be converted similarly **(0=AAAA, 1=AAAT, 2=AATT, 3=ATTT, 4=TTTT)** or **(0=1111, 1=1112, 2=1122, 3=1222, 4=2222)**. This is important as different analyses require different data formats, such as PolyRelatedness (Huang et al. 2016) or STRUCTURE (Pritchard et al. 2000). **Note:** Missing data is encoded as -9.

Basic data analysis

Calculating allele frequencies

This data set contains a large proportion of homozygous alleles, so it is recommended to test if there is a large proportion of minor alleles which may bias the downstream analysis and which need to be filtered prior to proceeding. For calculating minor allele frequency (maf), in this study we opted for weighing the genotypes by ploidy to avoid a ploidy bias.

```
gt.comb <- rbind(gt2x,gt4x) #combine the two data sets after ensuring the number of loci matches

gt.comb[gt.comb==-9] <- NA #replace missing data value (-9) with NA
```

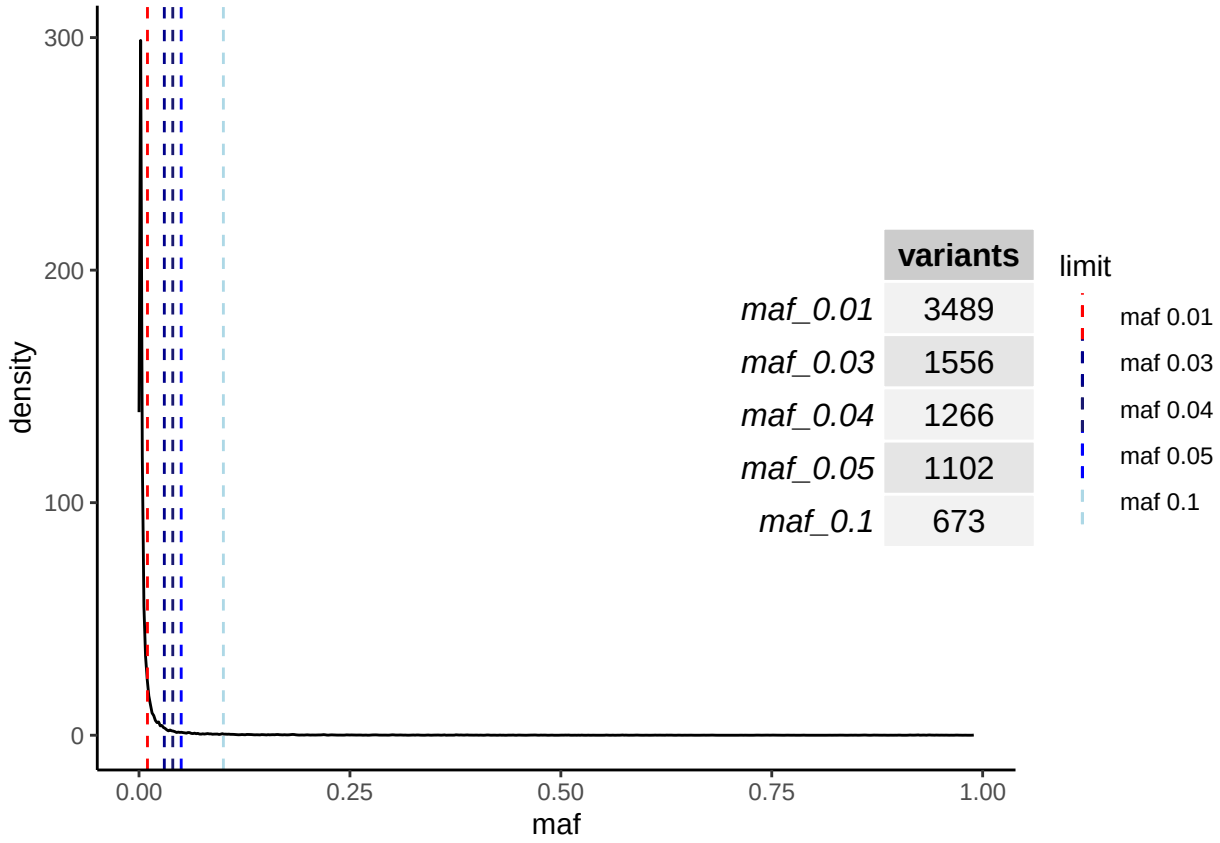
```r
#Form sum across columns (=over all individuals), and do not account for NA; divide by the number of di

maf.weighted <- data.frame("maf"=colSums(gt.comb, na.rm = T)/((nrow(gt2x)*2)+nrow(gt4x)*4))
```

After calculating allele frequencies, the number of remaining variants for each maf threshold can be summarized.

```r
maf.summ <- maf.weighted %>%
  summarize(maf_0.01=sum(maf>0.01),
            maf_0.03=sum(maf>0.03),
            maf_0.04=sum(maf>0.04),
            maf_0.05=sum(maf>0.05),
            maf_0.1=sum(maf>0.1)) %>%
  t()

colnames(maf.summ) <- "variants"
print(maf.summ)
```

```
##          variants
## maf_0.01     3489
## maf_0.03     1556
## maf_0.04     1266
## maf_0.05     1102
## maf_0.1       673
```

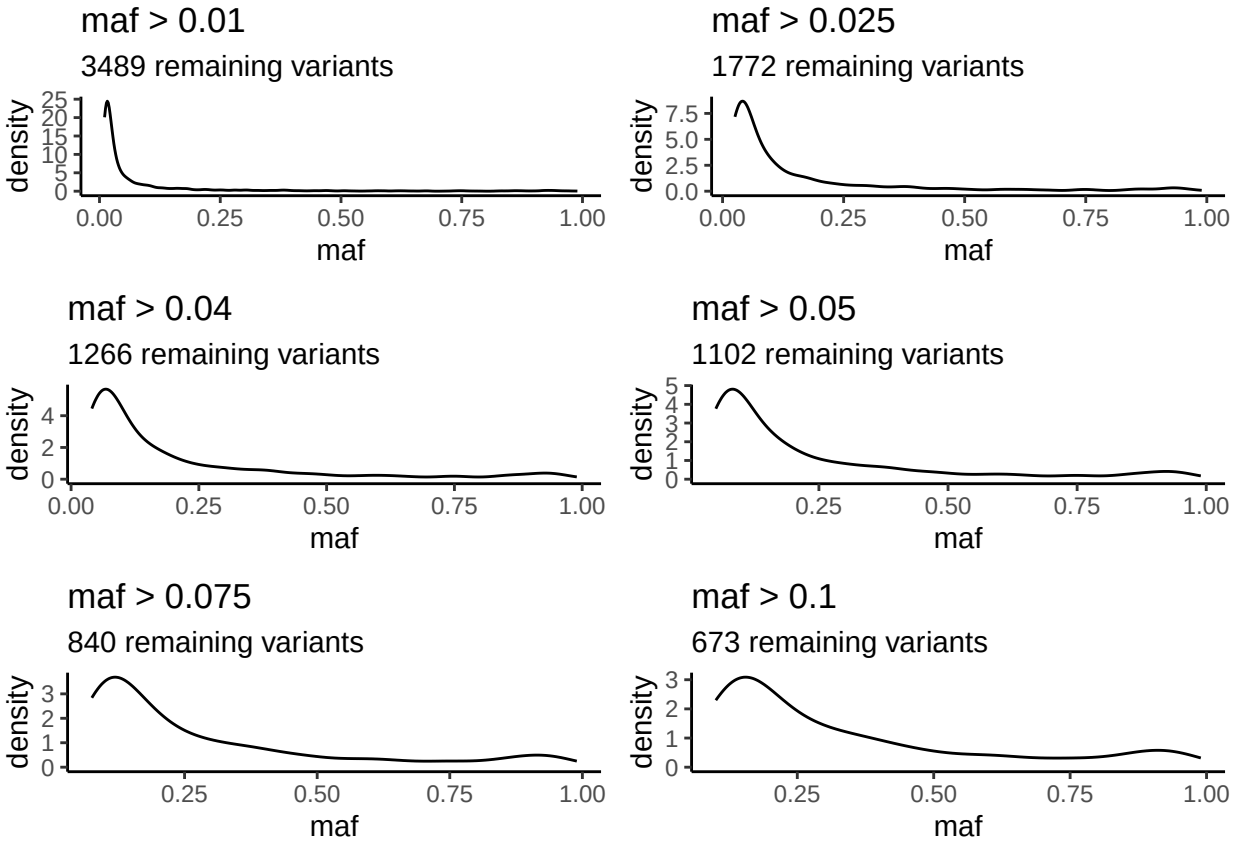
Additional visualization of the distribution of allele frequencies and marking of thresholds (such as 0.01, 0.05, 0.1) may further facilitate understanding of the data set.




```
## Saving 6.5 x 4.5 in image
```

1. The plot above shows that there is a high amount variants with frequencies lower than the thresholds, which could be explained with the high amount of PCR cycles during the ddRAD library preparation or the sensitivity of the genotyper.
2. Some allele frequencies reach values >0.5. This is because GATK encodes variants as homozygote reference (0/0), heterozygote (0/1) and homozygote (1/1) alternate. EBG then takes these values assuming that the reference represents the major allele, while the reference can also contain rare alleles. To this end, we define our allele frequencies as reference and alternate alleles for consistency with GATK.

After filtering for different thresholds, the visualization of the allele frequencies allows to assess distribution and number of remaining SNPs.



This study opted to choose a threshold of 0.04 (4%), as this represents around 18 individuals which corresponds well to the initial number of samples per plot (20) and allows to pick up signals which may be unique for a single plot.

To prepare the data for different analyses, the data set needs to be complemented with sample info (names, ploidy), the position of the variants and the genotypes. This combined data set can then be filtered and converted to formats as required by different analysis software.

Filtering allele frequencies

In this step, the previously calculated allele frequencies serve as a basis for filtering variants. The calculated frequencies are appended to each variant position inferred by EBG and variants below a certain threshold are removed.

```
vars.maf <- cbind(shared.vars, maf.weighted) # append the allele frequencies to the shared variants for

names(vars.maf) <- c("CHROM", "POS", "maf") # rename to avoid incompatibilities later on
```

```
write.csv(vars.maf, "results/tables/shared_vars_maf.csv", row.names = F, quote = F)
```

The variant positions with the corresponding frequencies can be saved, as it may be helpful for future analysis.

```
maf.filter <- 0.04 # set here the best suitable filter based on the figures before
#we can  re-use the previously calculated "maf_weighted" object. as maf is calculated per locus, we can

# custom function located in the source file to filter for a specific maf
gt.comb.filtered <- filter_maf(gt.comb, vars.maf, maf.filter)
```

```
## [1] "1266 Variants left after filtering for maf > 0.04"
```

```
ind.comb <- rbind(ind2x,ind4x)

names(ind.comb) <- c("Sample_ID","Ploidy")
data.comb <- cbind(ind.comb, gt.comb.filtered)
write.csv(data.comb, row.names = F, paste0("results/tables/SNPs_maf",maf.filter,".csv"))
```

The data set is now filtered data set that contains no variant with a an alternate allele frequency of < 0.04. In this set, 1266 remain.

Data conversion

Downloading PolyRelatedness

The program PolyRelatedness is available for most operating systems, which is convenient as a bash/R link can be used to contain the command for execution within R to maintain reproducibility and avoid switching between platforms.

The program can be downloaded (in this case to a `Software` directory located outside of the R project) and unzipped with R, though I recommend extracting it externally as the R unzip() function on MacOS detects a unsupported character in one of the references listed within the program directory.

```
PolyRelatedness <- "https://github.com/huangkang1987/polyrelatedness/raw/master/PolyRelatedness_1.11b.z

#path <- "path/to/software/dir"
path <- "~/Software/"

download.file(PolyRelatedness, paste0(path,"PolyRelatedness_1.11b.zip"))
unzip(zipfile = paste0(path,"PolyRelatedness_1.11b.zip"))
```

Preparing data for relatedness analysis (PolyRelatedness)

The software PolyRelatedness (Huang et al. 2014) is suitable for relatedness analysis of polyploids and requires a sample column (for pairwise individual relatedness), a population column (for population relatedness) and the genotypes encoded as alleles, such as 11,12,1222 etc., and missing data is expected to be 0 (or a positive integer not otherwise present in the data set).

The function recode2polyrel automatically detects ploidy (expandable up to octaploids) and converts it to such format.

```
polyrel.in <- recode2polyrel(data.comb)
```

```
## [1] "Dataset with 2 different ploidy levels detected. Converting accordingly..."
## [1] "211 2x individuals and 1266 variant sited detected. Converting..."
## [1] "227 4x individuals and 1266 variant sited detected. Converting..."
```

Inspect whether the conversion worked:

```
head(polyrel.in[1:5])
```

```
##   Sample_ID Ploidy V1 V6 V30
## 1     C1_01      2 55 55  66
## 2     C1_02      2 56 55  55
## 3     C1_03      2 55 55  56
## 4     C1_04      2 55 55  66
## 5     C1_05      2 55 55  55
## 6     C1_06      2 55 55  66
```

```
tail(polyrel.in[1:5])
```

```
##     Sample_ID Ploidy   V1   V6  V30
## 433     U4_01      4 5555 5555 5556
## 434     U4_05      4 5555 5555 5566
## 435     U4_11      4 5555 5556 5566
## 436     U4_15      4 5555 5555 5555
## 437     U4_18      4 5555 5555 5556
## 438     U4_20      4 5555 5555 5666
```

Conversion worked well, now some samples need to be removed. This step **should** be done prior to data analysis, however this data set contained samples with ambiguous/uninformative ploidy. Confirmation of these samples was received after library preparation, thus normalization by dosage, and sequencing, for that reason those samples have not been removed earlier.

Explanation: O1_01 and O2_03 - potential triploid P3_02_rep and O3_03_rep - replicates genotyped as alternative ploidy due to low quality tissue for flow cytometry (confirmed afterwards) C4_08 and C3_19 - samples with wrong dosage after library preparation

Additionally, we can add a population column, which in the labeling scheme of our study are the first two characters of the sample ID.

```
rm.inds <- c("C4_08","O1_01", "P3_02_rep", "O3_03_rep", "O2_03", "C3_19")

polyrel_input <- polyrel.in %>%
  select(-Ploidy) %>%
  mutate(pop=substr(Sample_ID,1,2)) %>%
  relocate(pop, .after=Sample_ID) %>%
  filter(!grepl(paste(rm.inds,collapse="|"),.$Sample_ID))
```

After filtering ambiguous samples, the data set can be exported.

Data preparation PolyRelatedness

```
filename <- "all_maf004"

write_delim(polyrel_input, paste0("~/Software/PolyRelatedness_1/",filename), delim ="\t", quote = "none
```

Finally the genotype file for Polyrelatedness needs to be complemented with software configuration as explained in the PolyRelatedness manual (https://github.com/huangkang1987/polyrelatedness).

The header, containing the configuration, requires the number of digits for each allele, the number of output digits for the relatedness coefficients, encoding for missing- and ambiguous alleles as well as the number of threads to use. Additionally, an end of file statement is required.

As the header and footer files are not provided within PolyRelatedness, these files are provided here in the `data` directory and may be moved/copied to the directory which the PolyRelatedness software.

```
cat(readLines('~/Software/PolyRelatedness_1/header.txt'), sep = '\n')
```

```
//configuration
//#alleledigits(1~4)    #outputdigits(0~10) #missingallele  #ambiguousallele    #nthreads(1~64)
1   8   0   7   8
//genotype
```

As Unix users we can link R with bash to automatize the workflow and keep records of what is exported. Most conveniently, the header, genotype and footer files should to be in the same directory.

First we specify the command to combine the files, and then execute it with the R/bash link

```
outdir <- "~/Software/PolyRelatedness_1/"
outfile_name <- paste0(filename,"_polyrel_in.txt")

combine <- paste0("cat ", outdir,"header.txt ",outdir,filename," ",outdir,"end.txt > ",outdir,outfile_na
system(combine)
```

Similarly, the program can be executed using the R/bash link, required the directories are specified correctly.

Otherwise the program needs to be executed from the commandline as

./PolyRelatedness infile.txt outfile.txt e 5 0

where e designates estimating relatedness, 5 designates the ritland estimator and 0 indicated individual pairwise comparison. More info at: https://github.com/huangkang1987/polyrelatedness

```
polyrel_outfile <- paste0(outdir,filename,"_polyrel_out.txt")
command <- paste("~/Software/PolyRelatedness_1/PolyRelatedness",paste0(outdir,outfile_name),polyrel_outf
system(command,intern = T)
```

```
## [1] "Estimating allele frequency by EM algorithm.....................................................
## [2] "Allele frequency has been writed to 'freq_em.txt'.\r"
## [3] "Progress: \r"
## [4] "||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\r"
## [5] "\r"
## [6] "Done! \r"
## [7] "Results are saved in the output file.\r"
```

Data preparation STRUCTURE

An excellent method to assign samples to populations and visualizing the results is the software STRUCTURE (Pritchard et al. 2000). Structure requires a specific format to which the dataset needs to be converted. The heteroploid data set prepared above can be re-used and filter for individuals and allele frequencies.

```
data.comb.str <- data.comb %>%
  filter(!grepl(paste(rm.inds,collapse="|"),.$Sample_ID)) %>%
  select(c(-Sample_ID,-Ploidy)) # filter individuals/pops/etc.

S_combined_structure <- filter_maf(gt.comb, vars.maf, maf.filter, remove_columns = F) #use function to
```

```
## [1] "1266 Variants left after filtering for maf > 0.04"
```

As STRUCTURE requires the SNPs to be unlinked, only one SNP per (ddRAD)locus/bp-window needs to be extracted. As the genotyping pipeline used here does not provide (dd)RAD locus information, SNPs can be filtered by specifying the distance which they need to be at least apart. In this case, we use the maximum locus length (139bp). The select_singleSNP() function extracts one random variant per window size.

```
sSNP_S_combined_structure <- select_singleSNP.3(S_combined_structure, window_size = 139)# use select si
```

```
## [1] "Extracting a single SNP at least 139 bp apart"
## [1] "Extracted 194 SNPs for Chromosome CM042338.1"
## [1] "Extracted 149 SNPs for Chromosome CM042339.1"
## [1] "Extracted 154 SNPs for Chromosome CM042340.1"
## [1] "Extracted 156 SNPs for Chromosome CM042341.1"
## [1] "Extracted 80 SNPs for Chromosome CM042342.1"
## [1] "Extracted 161 SNPs for Chromosome CM042343.1"
## [1] "Extracted 124 SNPs for Chromosome CM042344.1"
## [1] "Extracted 107 SNPs for Chromosome CM042345.1"
## [1] "Extracted 130 SNPs for Chromosome CM042346.1"
## [1] "Extracted 2 SNPs for Chromosome JAKJXK010000010.1"
## [1] "Extracted 3 SNPs for Chromosome JAKJXK010001541.1"
## [1] "Extracted 540 single SNPs"
```

The distance between SNPs can be validated to ensure no selected SNP originated from the same locus.

```
sSNP_S_combined_structure %>%
  group_by(CHROM) %>%
  mutate(diff=POS-lag(POS)) %>%
  select(CHROM, diff) %>%
  summarize(min_dist=min(diff, na.rm=T)) %>%
  head(9)
```

```
## # A tibble: 9 x 2
##   CHROM      min_dist
##   <chr>         <dbl>
## 1 CM042338.1      696
## 2 CM042339.1    26105
## 3 CM042340.1     2595
## 4 CM042341.1    24597
## 5 CM042342.1   612330
## 6 CM042343.1     2218
## 7 CM042344.1   146303
## 8 CM042345.1     9786
## 9 CM042346.1      929
```

Some of the variants may still be genomically linked, but as the purpose here was to select a single SNPs per RAD locus, this is an acceptable result to continue and convert to STRUCTURE format. If information of linkage blocks is known, the window size can be adjusted accordingly.

```
sSNP_S_combined_structure_t <- sSNP_S_combined_structure %>%
  ungroup %>%
  select(c(-CHROM,-POS, -maf,-Locus)) %>%
  t() #remove unwanted columns and transpose data set so it's compatible with Structure (i.e. ind in ro

###now convert to structure genos, which requires each allele of an individual in a row and each locus

sSNP_S_combined_structure_t <- cbind(ind.comb, sSNP_S_combined_structure_t) # combine name of samples w

data.comb.str <- sSNP_S_combined_structure_t %>%
  filter(!grepl(paste(rm.inds,collapse="|"),.$Sample_ID)) #%>% dplyr::rename("ploidy"="Ploidy")

sSNP_fin <- recode2structure(data.comb.str, as.pseudotetraploids = F) # utilize function to recode samp
```

```
## [1] "Dataset with 2 different ploidy levels detected. Converting accordingly..."
## [1] "206 2x individuals and 542 variant sited detected. Converting..."
## [1] "diploid genotypes detected...\n encoding diploids with missing alleles"
## [1] "224 4x individuals and 542 variant sited detected. Converting..."
## [1] "Polyploid genotypes detected...\n encoding as polyploids"
```

Evaluate if the data conversion worked as desired:

```
head(sSNP_fin[1:5],5)
```

```
## # A tibble: 5 x 5
##   Sample_ID `1`   `2`   `3`   `4`
##   <chr>     <chr> <chr> <chr> <chr>
## 1 C1_01     1     2     -9    1
## 2 C1_01     1     2     -9    2
## 3 C1_01     -9    -9    -9    -9
## 4 C1_01     -9    -9    -9    -9
## 5 C1_02     1     1     1     1
```

The diploids are encoded as tetraploids with missing alleles as recommended by Stift et al. (2019)

```
tail(sSNP_fin[1:5],5)
```

```
## # A tibble: 5 x 5
##   Sample_ID `1`   `2`   `3`   `4`
##   <chr>     <chr> <chr> <chr> <chr>
## 1 U4_18     1     2     1     1
## 2 U4_20     1     1     1     1
## 3 U4_20     1     2     1     1
## 4 U4_20     1     2     1     1
## 5 U4_20     1     2     1     1
```

Tetraploids are encoded with each allele in a row as expected.

```
#Let's see what we got for our structure analysis - we can also write these results as table as structu
dataset_dims <- dim(sSNP_fin)
filename <- paste0("structure_",dataset_dims[1]/4,"ind",dataset_dims[2]-1,"SNPs")
print(paste("In the final structure data set are",dataset_dims[1]/4,"Individuals and", dataset_dims[2]-
```

```
## [1] "In the final structure data set are 430 Individuals and 540 Variants"
```

```
str.info <- data.frame(nr_ind=dataset_dims[1]/4,
                       nr_vars=dataset_dims[2]-1)
write.table(str.info, paste0("results/tables/",filename,".info.txt"), row.names = F, quote=F)

#filename <- "ebg_structure004_correct_sSNP"
write.table(sSNP_fin, file= paste0("results/tables/",filename,".str"), quote=F, col.names = F, sep = "\
```

The exported .str file can be used for structure analysis using the program STRUCTURE. The .info.txt file contains the final number of individuals and SNPs in the dataset which need to be added to the STRUCTURE mainparams file. **Note**: This dataset does not contain a population column or any other extra column. This needs to be adjusted in the STRUCTURE mainparams file.

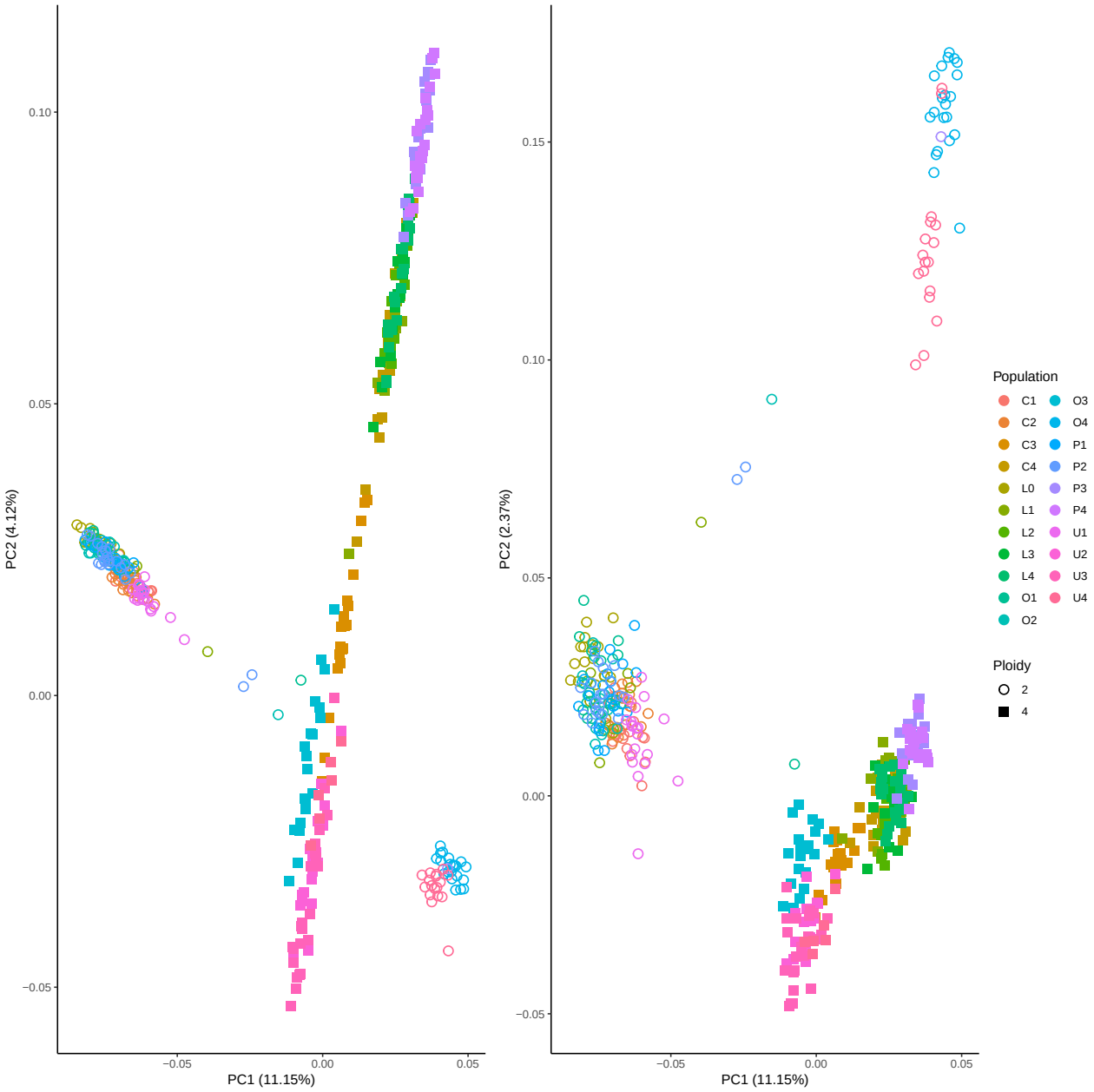
Data analysis

Analysing PolyRelatedness (PCoA)

The first analysis can be performed by extracting the eigenvalues from the matrix and plot it as principal coordinate analysis. In this example, ploidy is plotted as different shapes and the populations in different

colors. This can later be changed to i.e. STRUCTURE inferred populations.

```
polyrel_out <- read.table(polyrel_outfile, header=T, skip=7)

ind <- as.data.frame(row.names(polyrel_out))
colnames(ind) <- "Sample_ID"
sample_info <- merge(ind, ind.comb, by="Sample_ID", sort = F)
#extract eigenvalues from matrix
eigen <- eigen(polyrel_out)
#extract eigenvalues
ev <- NULL
for (i in 1:5){
  ex_var <- round(eigen$values[i]*100/sum(eigen$values),2)
  ev <- cbind(ev,ex_var)
}



pcoa <- as.data.frame(eigen$vectors)
pca_combined <- cbind(sample_info, pcoa)

pc12 <- pca_combined %>% mutate(Ploidy=as.factor(Ploidy), Population=as.factor(substr(Sample_ID,1,2))) %
  ggplot(aes(x=V1,y=V2, pch=Ploidy, col=as.factor(Population)))+
  geom_point(stroke=.8, size=3)+
  theme_classic()+
  labs(x=paste0("PC1 (", ev[1],"%)"), y=paste0("PC2 (", ev[2],"%)"))+
  scale_shape_manual(values=c(1,15))+
  labs(col="Population")

pc13 <- pca_combined %>% mutate(Ploidy=as.factor(Ploidy), Population=as.factor(substr(Sample_ID,1,2))) %
  ggplot(aes(x=V1,y=V3, pch=Ploidy, col=Population))+
  geom_point(stroke=.8, size=3)+
  theme_classic()+
  labs(x=paste0("PC1 (", ev[1],"%)"), y=paste0("PC2 (", ev[3],"%)"))+
  scale_shape_manual(values=c(1,15))+
  labs(col="Population")


ggpubr::ggarrange(pc12, pc13, common.legend = T, legend = "right")
```




```
ggsave(plot = pc12, path="results/figures/", filename = "PCoA12.pdf", device = "pdf")
```

There is clear separation both between and within diploid and tetraploid samples. Within the diploid samples, the Oregon and Utah samples are distinct from the Idaho samples. Within the tetraploids there is a similar separation between the Oregon and Utah samples from the Idaho samples.

Analysing PolyRelatedness (Heatmap)

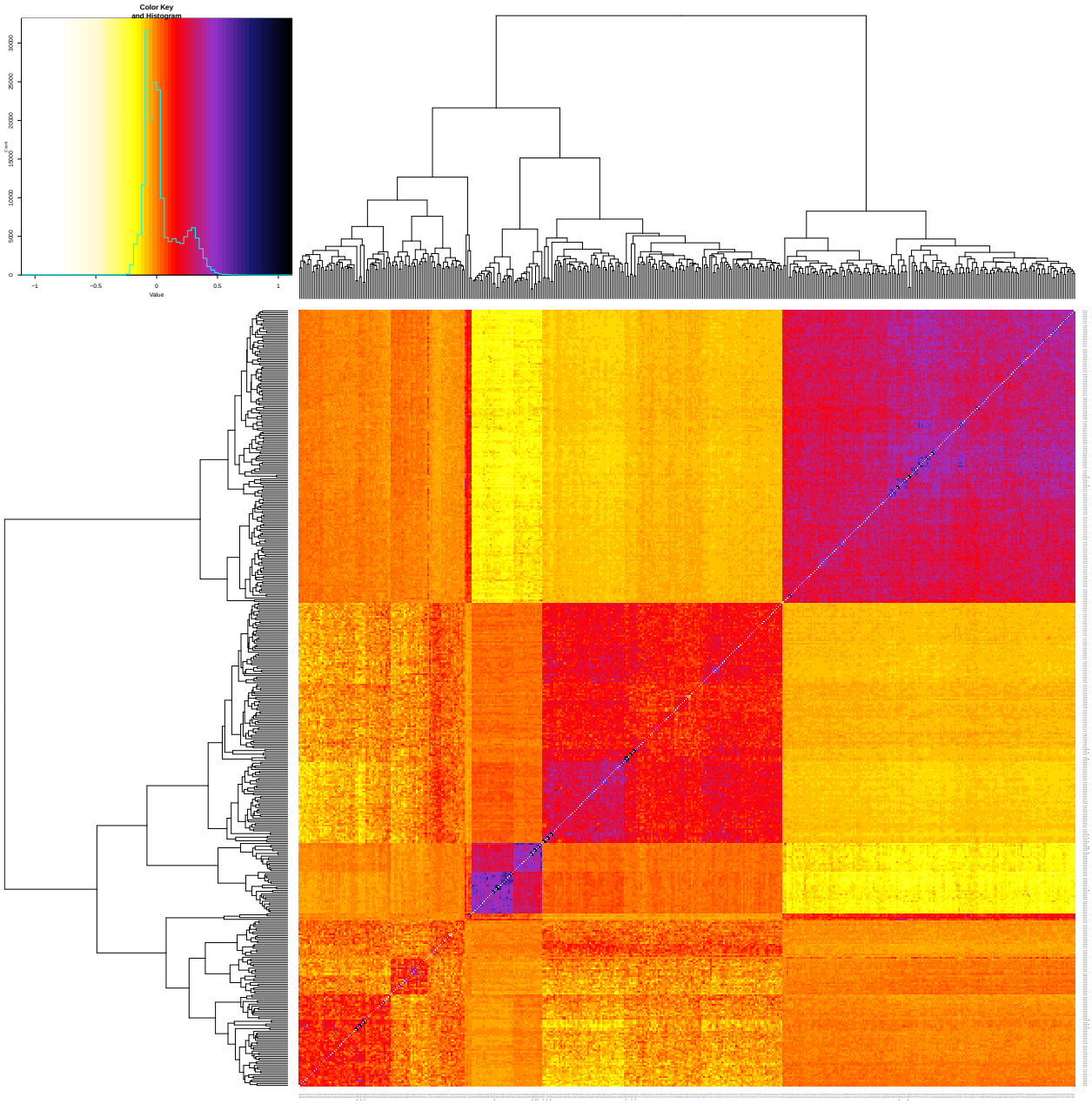
A heatmap is well suited to visualize the maximum of information as generated by PolyRelatedness.

We read in the relatedness file, replace the diagonal (i.e. individual compared with itself) by NA for better contrast. For clustering we use a dendrogram which uses hclust as underlying clustering algorithm.

```
polyrel_out <- read.table(polyrel_outfile, header=T, skip=7)
relatedness_matrix <- as.matrix(polyrel_out) # remove diagonal (i.e. individual compared with itself) f
diag(relatedness_matrix) <- NA # remove diagonal values
```

```
ind <-  row.names(polyrel_out)
heatmap_name <- substr(filename,1,nchar(filename)-4)
#map with dendrogram
```

```
heatmap.2(relatedness_matrix, trace="none", cexRow=0.2,
          cexCol = 0.2, labRow = ind,
          labCol = ind,
          col= colorRampPalette(c("white",
                                  "white",
                                  "lemonchiffon",
                                  "yellow",
                                  "red",
                                  "darkorchid",
                                  "midnightblue", "black"))(70))
```

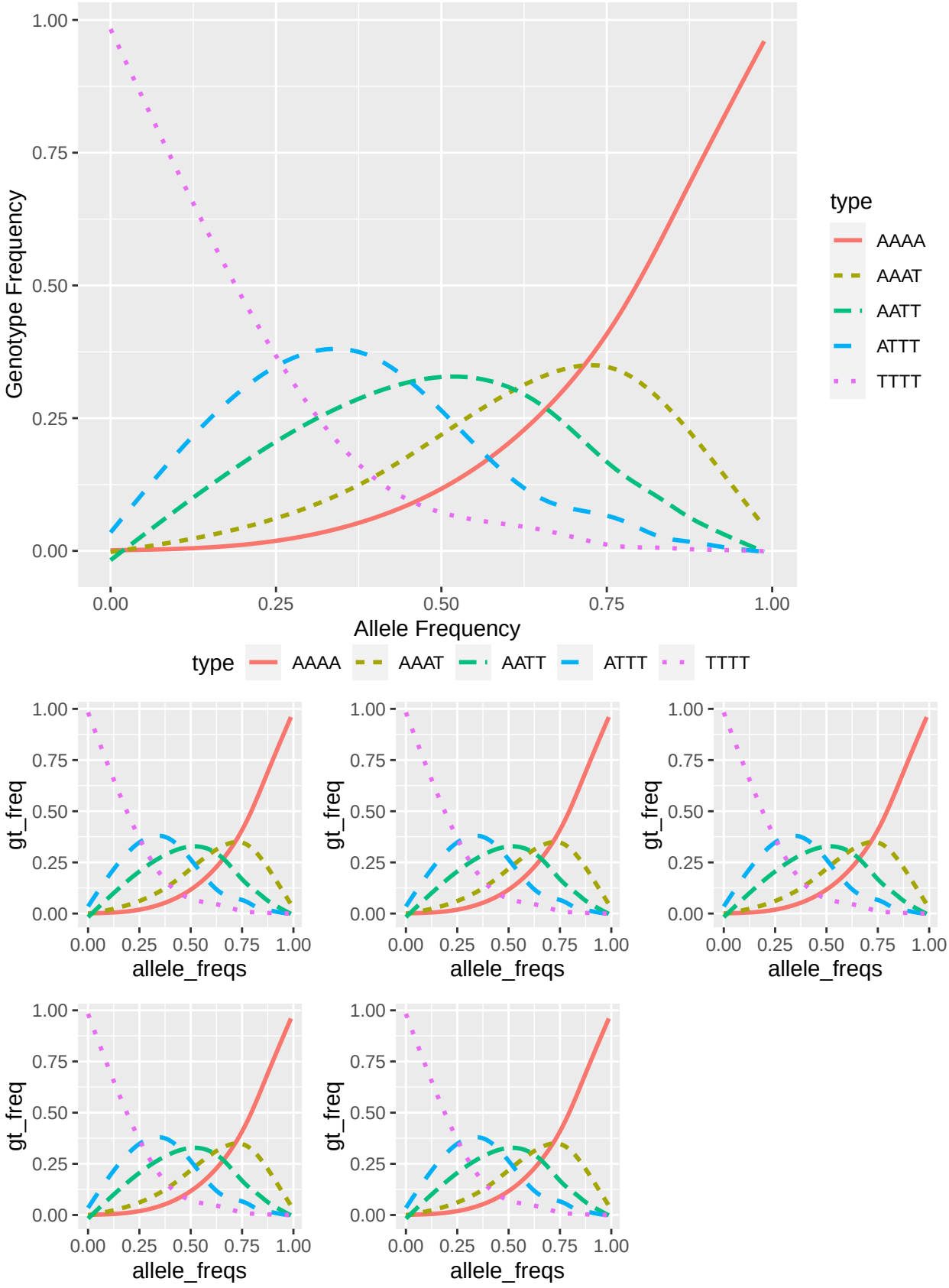


The heatmap shows the pairwise relatedness between individuals. In this case, the upper rigt block corresponds to *A. tridentata* ssp. *vaseyana* (mountain big sagebrush) samples, they appear to be very closely related and genetic diversity is high, and there is no apparent population structure. The central block correspond to *A. tridentata* ssp. *wyomingensis* tetraploids, most of those originating from the Idaho sites. Lower left of this central block there are the *A. tridentata* ssp. *tridentata* (basin big sagebrush) samples, which are furthermore split up by their location (i.e. Utah and Oregon). The lower left corner corresponds to *A. tridentata* ssp. *wyomingensis* tetraploids from Utah and Oregon.

The underlying matrix can further be used to calculate isolation by distance.

Identifying mode of inheritance

As a last step the maf filtered data set can be utilized to infer allele vs. genotype frequencies which can indicate the mode of inheritance.


Genotype Frequency
Allele Frequency
type
AAAA
AAAT
AATT
ATTT
TTTT
gt_freq
allele_freqs
0.00
0.25
0.50
0.75
1.00

Confirmation of data generation

```
## -- data
##    |__EBG_data
##       |__dipl_fin-genos.txt
##       |__dipl_indnames
##       |__shared-variants.txt
##       |__tetra_fin-genos.txt
##       |__tetra_indnames
##    |__PolyRelatedness_data
##       |__end.txt
##       |__header.txt
##       |__sagebrush_example.txt
##    |__STRUCTURE_data
##       |__K4_assignments.txt
## -- README.md
## -- results
##    |__figures
##       |__maf_distr_filtered.pdf
##       |__maf_distr_overall.pdf
##       |__PCoA12.pdf
##       |__structure_430ind540heatmap_combined.pdf
##    |__tables
##       |__shared_vars_maf.csv
##       |__SNPs_maf0.04.csv
##       |__structure_430ind540SNPs.info.txt
##       |__structure_430ind540SNPs.str
## -- src
##    |__EBG_pipeline.R
##    |__functions.R
```

All desired data and figures were generated. elapsed for running this pipeline.

Acknowledgements

I want to thank Ovidiu Paun, who provided information and helpful discussions about heteroploid data analysis which largely contributed to developing this pipeline.

Funding statement

This publication was made possible by the NSF Idaho EPSCoR Program and by the National Science Foundation under award number OIA-1757324. Additional funding was provided by the University of Idaho Stillinger Trust Expedition Fund, and the USDA Rocky Mountain Research Station.

References

Blischak PD, Kubatko LS, Wolfe AD. (2018) SNP genotyping and parameter estimation in polyploids using low-coverage sequencing data. Bioinformatics. 1;34(3):407-415. doi:10.1093/bioinformatics/btx587.

Huang K, Ritland K, Guo S, Shattuck M, Li B. (2014) A pairwise relatedness estimator for polyploids. Mol Ecol Resour. 2014 Jul;14(4):734-44. doi:10.1111/1755-0998.12217.

Pritchard JK, Stephens M, Donnelly P. (2000) Inference of Population Structure Using Multilocus Genotype Data, Genetics, Volume 155, Issue 2, 1 June 2000, Pages 945–959, https://doi.org/10.1093/genetics/155.2.945

Ritland, K. (1996). Estimators for pairwise relatedness and individual inbreeding coefficients. Genetics Research, 67(2), 175-185. doi:10.1017/S0016672300033620

Stift, M., Kolář, F., and Meirmans, P. G. (2019). Structure is more robust than other clustering methods in simulated mixed-ploidy populations. Heredity 123, 429–441. doi:10.1038/s41437-019-0247-6.

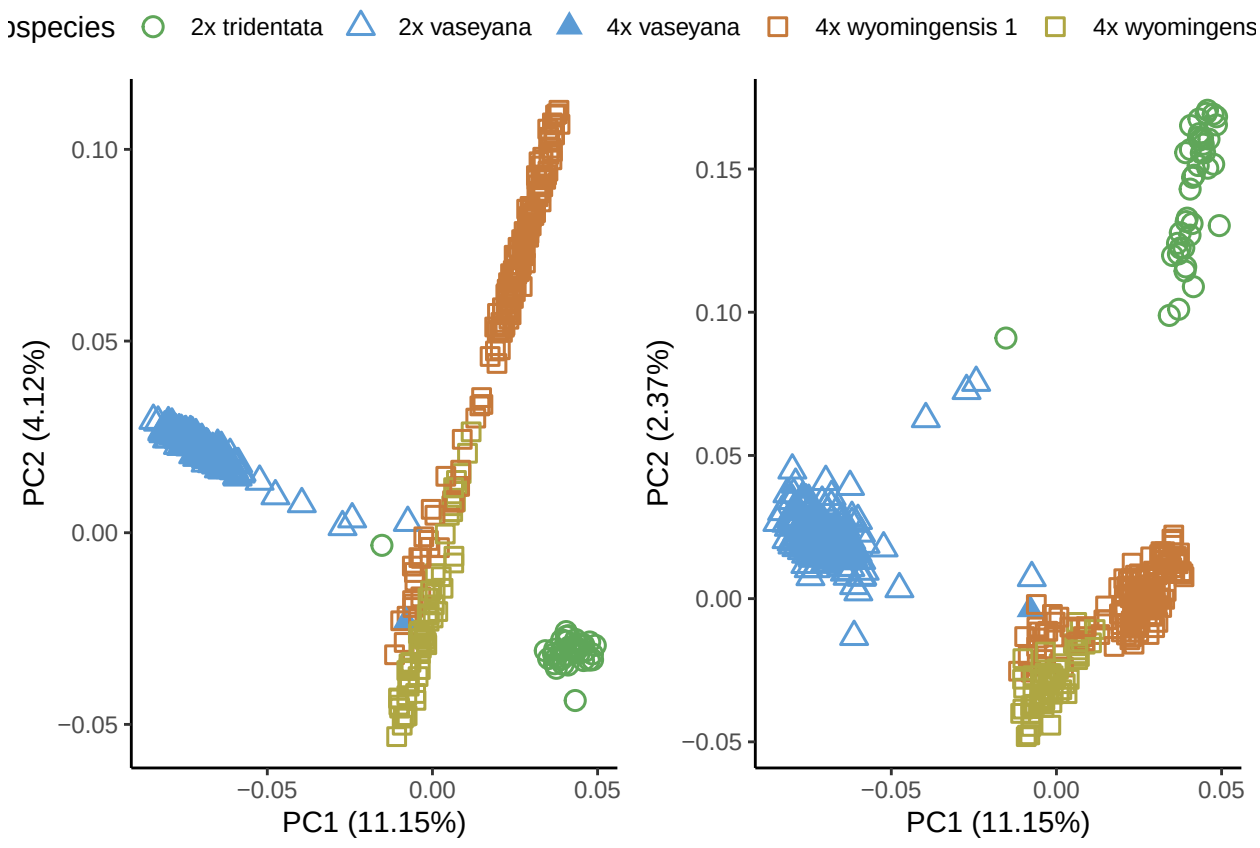
Appendix: Publication ready plotting

PCoA

To color the samples by their STRUCTURE based subspecies assignments, a file containing the ancestry proportions and the assigned subspecies based on these proportions can be utilized. The K=4 STRUCTURE assignments inferred by Grossfurthner et al. (2023) are provided in the data directory as `K4_assignments.txt`. The subspecies have been assigned by the max. ancestry proportion.

```
ssp.assignments <- read.table("data/STRUCTURE_data/K4_assignments.txt", header=T)

assignments.reformat <- ssp.assignments %>%
  dplyr::mutate(Sample_ID = ifelse(grepl("NA", Sample_ID),
                                   paste0(substr(Sample_ID,1,2), "_",
                                          substr(Sample_ID,3,4),"_rep"),Sample_ID))


pca_combined <- cbind(sample_info, pcoa) %>%
  left_join(., assignments.reformat, by="Sample_ID")

pca_combined <-pca_combined %>%  mutate(ssp_pl=paste0(ssp,"_",Ploidy, "x"))

shapes_pca <- c("vas1_2x"=2,
            "vas1_4x"=17,
            "hybrid"=10,
            "tri_2x"=1,
            "tri_4x"=16,
            "wyo1_2x"=0,
            "wyo1_4x"=22,
            "wyo2_4x"=22)
cols_pca <- c("vas1_2x" = "#5b9bd5", # darkblue
              "vas1_4x" = "#5b9bd5", # lightblue
              "tri_2x" = "#5fa659",
              "tri_4x" = "#5fa659",# green
              "wyo1_2x" = "#c6793a", # brown
              "wyo1_4x" = "#c6793a", # brown
              "wyo2_4x" = "#aea642")


labels <- c("vas1_2x"="2x vaseyana",
            "hybrid"="hybrid",
            "vas1_4x"="4x vaseyana",
            "tri_2x"="2x tridentata",
            "wyo1_4x"="4x wyomingensis 1",
            "wyo2_4x"="4x wyomingensis 2")
name <- "subspecies"


pc12 <- pca_combined %>% mutate(Ploidy=as.factor(Ploidy), Population=as.factor(substr(Sample_ID,1,2))) %
  ggplot(aes(x=V1,y=V2, pch=ssp_pl, col=ssp_pl))+
  geom_point(stroke=.8, size=3)+
  theme_classic()+
```

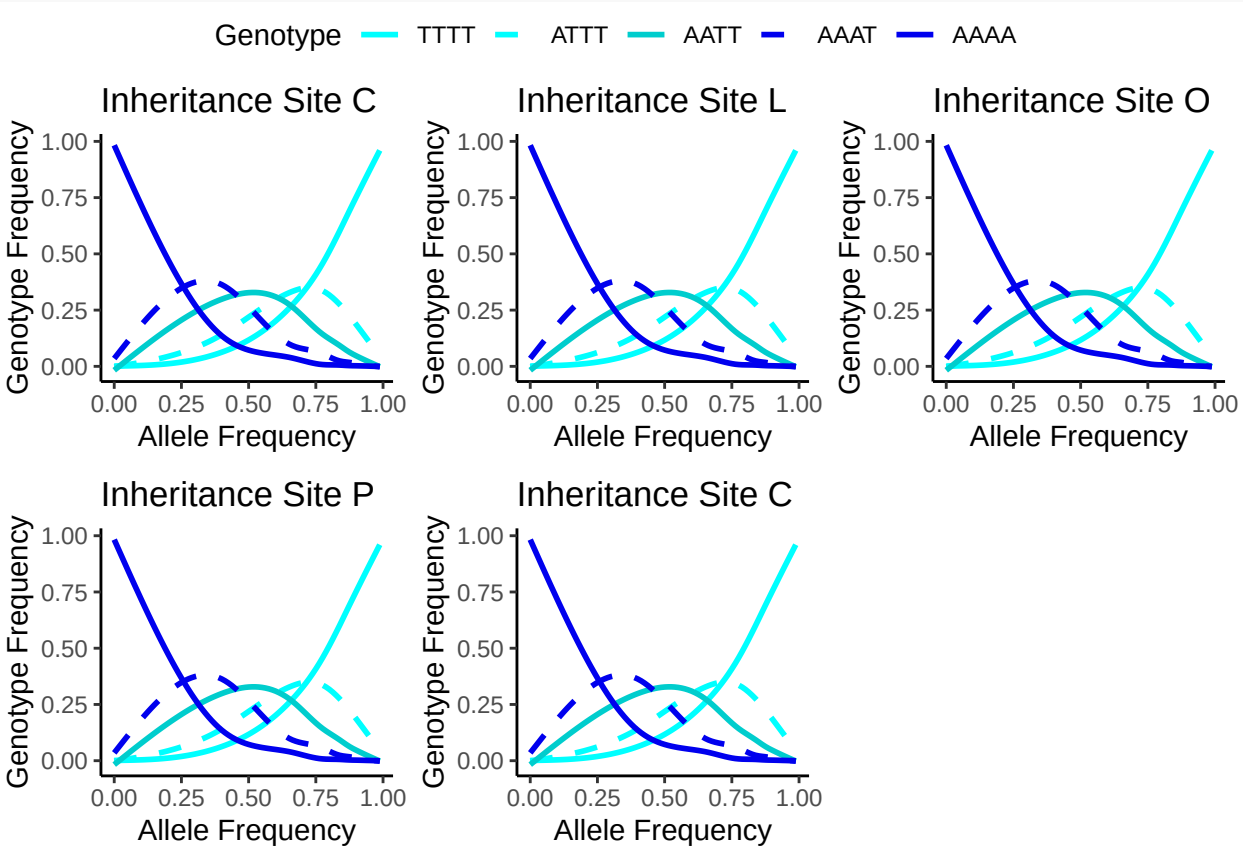
```
  labs(x=paste0("PC1 (", ev[1],"%)"), y=paste0("PC2 (", ev[2],"%)"))+
  scale_color_manual(values=cols_pca, limits=force, name=name, labels=labels)+
  scale_fill_manual(values=cols_pca, limits=force, name=name, labels=labels)+
  scale_shape_manual(values=shapes_pca, limits=force, name=name, labels=labels)

pc13 <- pca_combined %>% mutate(Ploidy=as.factor(Ploidy), Population=as.factor(substr(Sample_ID,1,2))) %
  ggplot(aes(x=V1,y=V3, pch=ssp_pl, col=ssp_pl))+
  geom_point(stroke=.8, size=3)+
  theme_classic()+
  labs(x=paste0("PC1 (", ev[1],"%)"), y=paste0("PC2 (", ev[3],"%)"))+
  scale_color_manual(values=cols_pca, limits=force, name=name, labels=labels)+
  scale_fill_manual(values=cols_pca, limits=force, name=name, labels=labels)+
  scale_shape_manual(values=shapes_pca, limits=force, name=name, labels=labels)


ggpubr::ggarrange(pc12, pc13, common.legend = T)
```



Inheritance plots

```
tetraploids_site <- tetraploids %>%
  mutate(site=as.factor(substr(Sample_ID,1,1)))


for (i in levels(tetraploids_site$site)){
  site <- tetraploids_site %>%
    filter(site==i)
```

```
  estimates_site <- est_inheritance(tetraploids)

  p <- ggplot(estimates_site, aes(x = allele_freqs , y = gt_freq, group = type, color = type, linetype=
  geom_smooth(method='gam',se = F, fullrange=T)+
  theme_classic()+
  labs(title=paste("Inheritance Site", i))+
  scale_color_manual(values = c("TTTT" = "blue2",
                                "ATTT" = "blue2",
                                "AATT"="cyan3",
                                "AAAT"="cyan", "AAAA"="cyan"),
                     labels=rev(c("AAAA","AAAT", "AATT", "ATTT", "TTTT")), name="Genotype")+
  scale_linetype_manual(values=c("TTTT" = "solid",
                                 "ATTT" = "dashed",
                                 "AATT"="solid",
                                 "AAAT"="dashed",
                                 "AAAA"="solid"),
                        labels=rev(c("AAAA","AAAT", "AATT", "ATTT", "TTTT")), name="Genotype")+
  labs(y="Genotype Frequency", x="Allele Frequency")



  inheritance_plots[[i]] <- p
}


ggpubr::ggarrange(inheritance_plots$C,inheritance_plots$L, inheritance_plots$O, inheritance_plots$P,inh
```



```
for (i in 1:length(inheritance_plots)){
  pdf(paste0("results/figures/inheritance_plot",names(inheritance_plots)[i],".pdf"))
  print(inheritance_plots[i])
  dev.off()
}
```