MANUAL

**SambaR**

Snp datA Management and Basic Analyses in R

**Menno de Jong**

HoelzeL Lab, Biosciences Department, Durham University, Durham, UK

AG Janke, BIK-F, Senckenberg Institute, Frankfurt, Germany

*Latest update: 22-09-2020 mennodejong1@hotmail.com*

SambaR is a R package for automated SNP dataset management and population genetic analyses. Whilst developing SambaR I was working on datasets generated by the ddRADseq protocol, but this package can also be used to analyse SNP datasets generated with other protocols, including whole genome resequencing. Sambar accepts biallelic SNPs only.

Because my datasets contained information about deer populations, I decided to name the package after a deer species called sambar (*Rusa unicolor*). But maybe an easier way to remember the name is that this package makes analyses of SNP datasets so easy and straightforward that you feel like wanting to dance the samba – or so I hope.

Please don’t be scared by the number of pages this manual contains. To run the main analyses with default settings, you only have to go through sections 1 to 7, which will be quick and quite possibly well worth your time. My aim is to help you to get a wide range of standard analyses (e.g. quality control, PCA, CA, DAPC, MDS, Nei’s D, Fst, HWE, SFS, LD, theta, Tajima’s D, inbreeding, admixture and selection analyses) done and your plots made with a minimum number of commands, without first having to spend weeks or months to get acquainted with all kinds of software. This way you can focus on what is really important: your study questions.

The functions in this R package depend heavily on other R packages. I make no excuse for that, because SambaR is meant to streamline the workflow. It does mean however that except for citing R and SambaR, you also need to cite many other packages. A Bibtex library of the citations of (most) of these packages is included in the SambaR download. This library can be exported into any referencing software, such as Endnote or Zotero.

SambaR comes with absolutely no warranty. You are free to share and distribute the package and to make changes to the source script as you please.

***Run time and data size limitations***

SambaR has been tested on datasets of up to hundreds of individuals and millions of SNPs. However, the memory capacity of your computer might pose restrictions and might cause SambaR to run into a memory error. The following table might help you to assess whether SambaR is capable of processing your data on your computer:

*Run time of SambaR (excluding installation of dependencies) for various sized SNP datasets*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | **Computer XL** | **Computer M** | **Computer S** |
| ***type*** | | | Linux HPC | Windows10 Intel Core™ processer i5-8500 CPU | Windows10 Intel Pentium processer CPU B940 @2 GHz |
| ***number of cores*** | | | 72 | 6 | 2 |
| ***RAM*** | | | 756 GB | 16 GB | 4 GB |
| ***data set*** | ***n\_snps*** | ***n\_inds*** |  |  |  |
| unpublished data | 100K | 85 | 1 hour | Memory Error | Memory Error |
| *Bosse et al. 2017* | 485K | 3015 | not tested | Memory Error | Memory Error |
| *Bosse et al. 2017* reduced | 485K | 100 | not tested | 4 hours | Memory Error |
| *Liu et al. 2014* reduced | 1.1M | 18 | not tested | 1 hour | Memory Error |
| *Viengkone et al. 2017* | 3K | 414 | not tested | <1 hour | 1 hour |
| *De Jong et al. 2020* | 50K | 92 | not tested | <1 hour | 1 hour |

***Output directories and files***

SambaR will generate within your working directory (the directory which contains your input files) a SambaR\_output directory with 7 subdirectories: Demography, Divergence, Diversity, Inputfiles, QC, Selection and Demography. After executing all functions, these subdirectories will contain input files, summary tables and ready to publish plots.

Every time you rerun a function, existing files from previous runs will be overwritten. Make sure that none of the plots are opened in a file viewer when rerunning a function, otherwise you will encounter errors.

SambaR creates most plots in up to 4 formats (i.e. eps, png, pdf, and wmf). Being vector-based, the pdf-files are suitable for stand-alone submission when your paper is accepted. The png and wmf files are of lower quality but are more easily imported into Word files, and therefore suitable to be included in Word files for review rounds. Keep in mind though that imported wmf-plots can cause problems when converting your word to pdf-format.

In principal pdf-files can be imported into Word files as well (see section ‘9. Manage your plots’ on how to do so), but these plots are heavy, and if you want to include many plots (for example supplementary documents), Word can crash.

# Content of this manual

[Content of this manual 3](#_Toc51708105)

[1. Do all at once 4](#_Toc51708106)

[2. Load SambaR 6](#_Toc51708107)

[3. Load dependencies 8](#_Toc51708108)

[4. Import your data into R 11](#_Toc51708109)

[4.1 Import biallelic data 13](#_Toc51708110)

[4.2 Import multiallelic data 21](#_Toc51708111)

[4.3 Convert from genlight object 22](#_Toc51708112)

[5. Observe your data 23](#_Toc51708113)

[6. Filter your data 25](#_Toc51708114)

[7. Analyse your data 30](#_Toc51708115)

[7.1 Population structure 31](#_Toc51708116)

[7.2 Population differentiation 35](#_Toc51708117)

[7.3 Genetic diversity 36](#_Toc51708118)

[7.4 Selection analyses 39](#_Toc51708119)

[7.5 Association analyses 44](#_Toc51708120)

[7.6 Population demography (ADDITIONAL ANALYSIS) 45](#_Toc51708121)

[8. Manage your data 50](#_Toc51708122)

[9. Manage your plots 53](#_Toc51708123)

[10. Understand your output 55](#_Toc51708124)

[10.1 The snps dataframe 56](#_Toc51708125)

[10.2 The inds dataframe 59](#_Toc51708126)

[10.3 SambaR calculations 62](#_Toc51708127)

[11. Simulate your data 81](#_Toc51708128)

[12. Solve or work around errors 85](#_Toc51708129)

# 1. Do all at once

Less than 10 R commands suffice to get almost everything done. These wrapper commands, which execute many analyses at once, roughly look like this (but you need to customize them, so keep on reading):

*source("C:/path/to/dir/SAMBAR.txt")*  section 2

*getpackages(myrepos='http://cran.us.r-project.org',mylib=NULL)* section 3

*setwd("C:/path/to/workdir/")* section 4

*importdata(inputprefix="yourprefix",sumstatsfile=TRUE,depthfile=TRUE)* section 4

*filterdata(indmiss=0.25,snpmiss=0.1,min\_mac=2,dohefilter=TRUE)* section 6

*findstructure(Kmax=6,legend\_pos="right",legend\_cex=2*,*symbol\_size=2)* section 7.1

*calcdistance(legend\_cex=2)* section 7.2

*calcdiversity(nrsites=NULL)* section 7.3

*selectionanalyses(do\_meta=TRUE,do\_pairwise=FALSE)* section 7.4

*backupdata(“my\_snpdata”)* section 8

The getpackages(), importdata() and filterdata() functions should be executed sequentially. Afterwards, you can run any of the remaining functions in the order you prefer. For example, if you are interested in doing selection analyses only, you don’t have to execute the findstructure(), calcdistance() and calcdiversity() functions.

If this is the first time you use SambaR, you will have to go through the manual to understand how to customize the commands, how to prepare the input files, and what to expect in terms of the output. No worries: as stated above, you will get through it quickly.

If you used SambaR before, you know what to do and you can simply fill in the correct paths and desired settings, and next copy paste the commands into R. For the first run, I recommend to copy paste the commands one by one rather than all together at once, in case you run into an error early on. But once you are sure there are no errors, you could run them all at once (or even put them in a function, so you would only have to run this single function).

**Additional analyses**

Some additional optional analyses require specific input files, and therefore need to be run independently. In this manual these analyses are labelled as ‘ADDITIONAL ANALYSIS’.

**Overview of workflow**

**Load SambaR into R**

*source("C:/path/to/dir/SAMBAR.txt")*

**Install and load dependencies**

*getpackages(myrepos='http://cran.us.r-project.org',mylib=NULL)*

**Import your data (plink format) into R**

*setwd("C:/path/to/workdir/")*

*importdata(inputprefix="yourprefix",sumstatsfile=TRUE,depthfile=TRUE,samplefile=NULL)*

**Filter your data**

*filterdata(indmiss=0.25,snpmiss=0.1,min\_mac=2,dohefilter=TRUE,* *snpdepthfilter=TRUE,min\_spacing=500)*

**population structure**

*findstructure()*

**population differentiation**

*calcdistance()*

**genetic**

**diversity**

*calcdiversity()*

**selection analyses**

*selectionanalyses()*

**Analyze your data**

# 2. Load SambaR

To load SambaR into R, first make sure that you use the most recent version of R. This can save you a lot of problems and frustration when SambaR attempts to install all required dependencies (see next section). Preferably also install the latest version of Rtools. This will take a few minutes only.

In theory SambaR might run without problems on fairly recent versions of R and you are welcome to give it a try. There are however no guarantees, as packages are being continuously updated.

Once you opened a new R session, load SambaR simply by typing:

*source("C:/ path/to/directory/SAMBAR.txt”)*

For example, if you have stored the source script on the C drive in the directory ‘ProjectA’ and the subdirectory ‘SNPanalysis’, you would type:

*source("C:/ ProjectA/SNPanalysis/SAMBAR.txt”)*

Note: there shouldn’t be any white spaces. Also note that R uses forward slashes rather than backward slashes.

***Trouble shooting***

One of the errors you might encounter is this one:

*Error: unexpected input in "source(“"*

In that case Word (or another text editor) has deformed the double quotes (from "" to “”). If so, either retype the double qoutes and run the command again, or copy paste via Notepad. This applies throughout this manual.

SambaR will tell you if it can find the file or path you specified. You can use the command ‘list.files()’ to see which files are present in any directory. For example, to see which files are present in the directory specified above, you would type:

*list.files(“C:/ ProjectA/SNPanalysis/”)*

# 3. Load dependencies

As mentioned before, SambaR depends heavily on other R packages (>2GB in total). Therefore, you need to install those as well. The good news is that SambaR comes with a function which automatically installs and loads all required packages. To execute this function, simply type:

*getpackages(myrepos='http://cran.us.r-project.org')*

If any of the dependencies still need to be downloaded, you will get a lot of rubbish on the screen, which is fine, as long as there are no errors. Warnings are allowed. You might for example get warnings about the R version under which a package has been built, or about objects which have been masked from packages.

If R asks you whether it should create a personal library, answer ‘y’ for yes (and answer again yes when it proposes a specific directory).

If R asks you whether you want to install from source the packages which needs compilation, answer ‘n’ for no (because that can take ages).

If R asks you whether it should update packages, answer ‘n’ for no (because that can take ages).

If R asks you to select a CRAN window, scroll down to ‘other windows’ and select any http window (rather than a https window).

The first time you run this function for a particular R version on your computer, the getpackages function will take several hours to complete. Subsequent executions will run in seconds.

**If you do not have root access**

If you don’t have root access, you need to specify the path to the directory where you want install the packages. For example, if you want to install the packages in the directory ‘home/userX/Rpackages’, you would type:

*getpackages(myrepos='http://cran.us.r-project.org',mylib=‘home/userX/Rpackages’)*

SambaR will automatically add the specified directory to the library paths, so that R knows where to find the installed packages.

**Trouble shooting**

When SambaR does not manage to install and load a certain package, the getpackages function will automatically halt and inform users about the problem. It will also export a control file called ‘mypackageslist.txt’. Users can edit this file to prevent SambaR from attempting to install certain packages, for example the ones which are problematic.

The control file classifies packages into three categories: ‘essential’, ‘recommended’, and ‘optional’. Essential packages are required for SambaR to run without errors. Recommended packages are needed for key analyses. If you run into a problem when trying to install an optional package, the easiest solution would be to simply omit this package. You can do so by setting it's value in the 'do\_install' column to FALSE.

***System Requirements***

Some of the packages which SambaR tries to install, most notably ‘sf’, depend on software tools which needs to be installed on your computer. Often these software tools are part of the standard installations, and you will not need to bother about it. But if they are not installed, you will run into an error similar to:

*Checking for sqlite… no*

*no*

*configure: error: sqlite not found or not executable*

*ERROR: configuration failed for package ‘sf’*

To prevent or solve this error, you need to have installed the following software:

sqlite

proj-devel-6.3.2-4.el8

proj-static-6.3.2-4.el8

geos-devel-3.7.2-1.el8

libxml2-devel

Installing is easiest to do if you have root access, using a command like:

*yum install sqlite*

Note however that, depending on your system, you might have to use an alternative to yum.

If you do not have root access yourself, ask your administrator to install the software.

***WARNING: Package not available***

If you do not have the most recent version of R, you might encountered the problem:

*WARNING: package not available for R version …*

If so, this probably means that the package depends on a more recent version of R. In that case you could either omit this package (by it's value in the 'do\_install' column of the ‘mypackageslist.txt’ file to FALSE), or, if the package is essential, install the latest R version and then try again.

***ERROR in readRDS(dest): error reading from connection***

The getpackages function will try to install Bioconductor packages. It can happen that you run into the error:

*ERROR in readRDS(dest): error reading from connection*

In that case one of the previous loaded packages causes the software to behave unexpectedly. To circumvent this problem, edit the ‘mypackageslist.txt’ file as follows: in the column ‘do\_install’ set all values to FALSE, except on the lines of Bioconductor packages. Open a new R session and run the getpackages() function again.

Once all Bioconductor packages have been installed, run the getpackages() function again with the do\_install column values set to TRUE.

***Problems with Rtools and github packages***

A few packages are installed from Github and therefore need a software called Rtools. If R can not find it, you will run into an error related to the following warning:

*WARNING: Rtools is required to build R packages, but is not currently installed. Please download and install Rtools custom from http://cran.r-project.org/bin/windows/Rtools/*

If so, download Rtools and afterwards rerun the getpackages() function.

# 4. Import your data into R

SambaR objects can be created either by importing data into R (section 4.1 and section 4.2) or alternatively by converting from an existing R genlight object (section 4.3).

If you want to import biallelic data (max two alleles per SNP), follow the instructions in section 4.1. If you want to import multiallelic data (more than two alleles per SNP), follow instructions in section 4.2. At present SambaR offers a very limited amount of analyses for multi allelic data.

**Is your data biallelic?**

If you generated your data using STACKS (RADseq data), your data is biallelic.

If you generated your data with a SNPcaller software as GATK, or Platypus, or samtools/bcftools mpileup you have ended up with a variant file with all kinds of variants. In that case you should filter out indels and multi-allelic SNPs, if you haven’t done so yet. You can do so for example with bcftools:

*BCFTOOLS view --exclude-types indels –max-alleles 2 -O z -o allsites.vcf.gz input.vcf.gz &*

**Should you filter your data before running SambaR?**

If you generated your SNP data with STACKS, there is usually no need to do additional filtering, meaning you can proceed directly from STACKS to SambaR. The reason is that STACKS applies filters, and that RADSEQ datasets are relatively small.

If, in contrast, you have whole genome resequencing data, you will need to filter your SNP data, and make sure to remove SNPs with low genotype quality and high proportions of missing data.

***Filter on genotype quality***

To filter on genotype quality and on combined read depth, you could for example use bcftools:

*BCFTOOLS filter --include "QUAL>=30 && DP>=800 && DP<=3000" input.vcf.gz -O z QC.vcf.gz &*

***Keep biallelic SNPs only***

You can also use bcftools to remove indels and SNPs with more than 2 alleles:

*BCFTOOLS view -include "AN>=96"* -*-exclude-types indels --max-alleles 2 -O z input.vcf.gz -o QC.vcf.gz &*

***Filter on levels of missing data***

You can also use bcftools to filter out SNPs with high levels of missing data. Say for example that you have a dataset of 50 diploid individuals, and you want to keep remove SNPs with more than 4 percent missing data, you could execute:

*BCFTOOLS view -include "AN>=96" -O z input.vcf.gz -o QC.vcf.gz &*

Alternatively, you can use PLINK, as will be explained below. You can also use PLINK (or other software) to filter on linkage disequilibrium. Having said that, SambaR uses a thinned dataset for structure analyses (by default keep only 1 SNP per stretch of 500 bp), which should in principle remove many SNPs in LD due to physical linkage.

As explained further on in this manual, you can use SambaR for quality control and for filtering your data on levels of missing data. However, SambaR, being an R package, has data size limitations. If you have big datasets (hundreds/thousands of individuals and/or >200K SNPs), this might lead to the error: ‘cannot allocate vector of size…’. One reason to remove SNPs with high levels of missing data prior to importing the data into R, is to prevent this error.

But this is not the only reason. Depending on the aims of your study, sacrificing low quality SNPs (which have high amounts of missing data) will not only avoid memory allocation errors, but also speed up the process, and, more importantly, lead to better results. A smaller dataset with high quality SNPs might provide more accurate results (e.g. more accurate picture of population structure and genetic diversity) than a bigger dataset with lower quality SNPs.

## 4.1 Import biallelic data

SambaR uses the ‘read.PLINK’-function of the Adegenet package to import biallelic genotype data into R. This function expects your files in RAW and BIM format, which are the binary versions of PED and MAP format. Here I will describe how to convert your data files into these formats, and how to subsequently import your data into R. These preparatory steps involve using software other than R, namely PLINK and vcftools and/or PGDspider.

The three steps to import your data into R: Software:

* Convert your data into a PED and MAP files plink2 or vcftools or PGDSpider
* Convert PED/MAP files to binary RAW/BIM files plink2
* Import RAW and BIM files into R. R

If you still need to install any of the programs listed above, here are the links:

<http://www.cmpg.unibe.ch/software/PGDSpider/>

<https://vcftools.github.io/downloads.html>

<https://www.cog-genomics.org/plink2/> (choose the stable beta version)

<https://cran.r-project.org/bin/windows/Rtools/>

If you have access to Durham University Hamilton cluster, you can run the programs from here:

*/ddn/data/fjsq43/Programs/fileconversion/PGDSpider\_2.1.0.3/PGDSpider2-cli.jar*

*/ddn/data/fjsq43/Programs/SNPanalysis/plinkv13jan2017/plink*

*/ddn/data/fjsq43/Programs/variantcalling/vcftools\_0.1.13/bin/vcftools*

**Convert from any format to PED/MAP**

There are many ways to create a PED/MAP file, depending on your starting point. Here I describe two options: convert from vcf format using vcftools, or convert from genepop format using PGDSpider.

***Option 1. Convert from vcf to PED/MAP***

Both plink2 and vcftools can quickly convert vcf files into ped/map format. I therefore recommend STACKS users to include the –V flag when running the STACKS populations command. (Another advantage of the vcf format and vcftools is that it can be used to calculate read depth, as explained below.)

To convert vcf to PED and MAP, type on the command line of your computer either:

*/path/to/plink –vcf inputfile.vcf –recode --out prefixoutputfile*

or:

*/path/to/vcftools --vcf inputfile.vcf --plink --out prefixoutputfiles*

Vcftools has the disadvantage that it does not like contig or chromosome names other than human chromosome numbers. You are therefore likely to see many warnings on the screen, stating: ‘Unrecognized values used for CHROM – Replacing with 0’. As a result, you will find that the first column of the MAP file, which should contain contig names, contains zero’s only. However, as vcftools uses the contig names to make SNP names (see column 2), this problem is potentially easy to fix. All you have to do is to insert in the following command the name of your MAP file (as highlighted in grey), and then execute this oneliner on the Linux command line:

*cut -f2 yourfile.map | cut -f1 -d ':' > mycontigs.txt && cut -f2,3,4 yourfile.map > mymap.txt && paste mycontigs.txt mymap.txt > yourfile.map && rm mycontigs.txt mymap.txt*

Normally the first column in PED files contains the names of your populations (and the second column the names of your samples), but as vcf files only store sample names and not population names, you will find that when converting from vcf to PED, both columns in the PED file contain the names of your samples. Therefore, you have to provide an additional population file. This should be a tab delimited file with two column called ‘name’ and ‘pop’. The first column contains the sample name and the second column contains the population name, like this:

*name pop pop2*

*sample1 pop1 subpop1*

*sample2 pop1 subpop2*

*sample3 pop2 subpop3*

*etc*.

The third column, ‘pop2’, is optional, and serves to define subpopulations within your main populations. For example: if you have samples from Europe, Asia and North America, the pop column could serve to assign samples to ‘Europe’, ‘Asia’ and ‘NA’, and the pop2 column could serve to assign samples to specific regions, e,g, MidEast and FarEast for Asia.

WARNING: population names may NOT contain underscores or spaces, because this will interfere with Sambar commands. In addition, because of plot margins, your population names may NOT contain more than 10 characters. Also note that populations should contain at least 2 or more individuals.

You will need this populations file when importing the data into R. But before you can import the data, you need to do one more step: convert from PED/MAP to RAW/BIM format.

Since you are on it, you can also calculate read depths per sample and SNP:

*/path/to/vcftools --vcf inputfile.vcf --depth &*

*/path/to/vcftools --vcf inputfile .vcf --site-mean-depth &*

***Option 2. Convert from genepop (or other formats) to PED and MAP using vcftools***

If your data is not in vcf format, don’t worry! You can use PGDspider to convert your data to a PED and a MAP file from almost any format. You can either run PGDspider using the graphical user interface (GUI), or run it from the command line. Note that the GUI of PGDspider has a data size limit, and therefore, depending on the size of your dataset, you might have no other option than to run PGDspider from the Windows command line.

If using the graphical user interface (GUI) of PGDspider, select as data input file ‘genepop’ and as data output file ‘PED’. Click on the button ‘create/edit SPID file’ , and select ‘SNP’ as dataset under the tab ‘Genepop – Parser Questions’. Under the tab ‘PED – Writer Questions’, answer ‘yes’ to the question whether you want to save an additional MAP file, and provide a name. This name should be the same as the name of the PED file. The extension is added automatically.

If running PGDspider from the Windows command line, you will first have to create the appropriate SPID-file (if you prefer, you can do so by using the graphical user interface), and afterwards execute on the command line a command which looks like this:

*java -jar PGDSpider2-cli.jar -inputfile infile –outputfile out.ped -spid spidfile.spid*

To save the time having to make SPID files yourself, I enclosed two SPID files needed for the conversion from either genepop or vcf format. The actual commands are:

*java -jar PGDSpider2-cli.jar -inputfile infile.genepop –outputfile out.ped -spid Genepop2PED.spid*

*java -jar PGDSpider2-cli.jar -inputfile infile.vcf –outputfile out.ped -spid VCF2PED.spid*

Both commands will output two files, called out.ped and out.map, to the working directory.

If you get empty output files (0 or 1 kb), one reason might be that PGDspider doesn’t like your population names (if different from pop). So it might help to indicate populations simply with the word ‘pop’ for now.

As the population names in the PED-file (first column) will be used by Sambar to name the populations, this moment (after the conversion) is a good moment to rethink how you want to name your populations, and possibly rename them. For example: if at the moment your PED-file lists your populations simply as ‘pop1’ and ‘pop2’ (etc), you might want to provide more telling names.

It is important to note that population names may NOT contain underscores or spaces, because this will interfere with Sambar commands. In addition, because of plot margins, your population names may NOT contain more than 10 characters. Note that populations should contain at least 2 or more individuals.

**Convert from PED/MAP to RAW/BIM format**

Now that you created your PED/MAP files, you might want to do a quick check. Does the number of lines in the PED file correspond with the number of individuals? Does the number of lines in the MAP file correspond with the number of SNPs?

If everything looks right, you are ready to move on to the next step. PED and MAP files of SNP datasets are generally too big to load directly into R. Therefore, you first need to compress your files from PED/MAP formats to RAW/BIM formats.

You can use PLINK to do so. The command, to be executed on either Unix or Windows command line, looks like this:

*plink --file prefix –-chr-set 95 –-allow-extra-chr --make-bed --recode A --out prefix*

The make-bed flag tells the software to make the bim file. The ‘recode A’ flag tells the software to create the raw file. In order for the PLINK command to work, the PED and the MAP file should have the same name (prefix), except for the extensions ‘.ped’ and ‘.map’. Use the –-allow-extra-chr flags to include snps located on contigs rather than on chromosomes. If not, you might end up with empty output files. The –chr-set indicates how many chromosomes are present at max (95 is unrealistically high for most species, but this is to prevent errors).

If you get the error ‘--file accepts at most 1 parameter’, or ‘unrecognized flag’, or ‘doesn’t accept parameters’, retype all the dashes and try again.

Whilst running the command, PLINK will output some information on the screen. Also here you could check whether the expected number of individuals and SNPs corresponds with the information on the screen. If you made sure your data is biallelic, you can avoid warnings about some SNPs being triallelic – this is probably due to an unexpected character for missing data.

The raw file is binary, but the bim file is not, so just to make sure everything went right you might want to compare it to the map file. (Just note: if you want to observe the files, open these files in a plain text editor, such as Notepad or Notepad++, and not in other programs such as Word, because this might corrupt the file.)

***Trouble shooting***

If your input file contains indels, PLINK will throw the error:

*Possibly irregular .ped line. Restarting scan, assuming multichar alleles.*

*Rescanning .ped file... 0%*

*Error: Half-missing call in .ped file at variant …, line 1.*

If so, redo the filtering steps (see above).

If your contigs (first column of the MAP-file) start with a number rather than a letter, PLINK assumes they are chromosomes, and might throw the error:

*Invalid chromosome ‘..’ on line ... of .map file.*

A solution would be to add the word ‘contig’ in front of the number. You can do so on the Linux command line with the following command:

*sed -i -e 's/^/contig/' inputprefix.map*

If the data is not properly sorted by chromosome/contig, PLINK might throw the error:

*Error: .bim file has a split chromosome. Use –make-bed by itself to remedy this.*

This means that your data might have, for example, some chromosome 5 markers, followed by chromosome 6 markers, and then followed again by chromosome 5 markers. Run the following command to sort the data:

*plink –file prefix –make-bed –out prefix.sorted*

If missing data is encoded by N instead of 0, PLINK might throw the error:

*Warning: Variant …. (post-sort) triallelic; setting rarest alleles missing.*

Note however that this error can also have other causes.

**Load RAW /BIM into R**

Now you are ready to import the data into R.

Open the R session in which you loaded Sambar and use the setwd() command to navigate to the directory on your computer where you stored your RAW/BIM files (and other optional input files):

*setwd("C:/Users/my/path/to/directory/”)*

Check whether your input files are present:

*list.files()*

If your input files are present, import your data using the function ‘importdata’:

*importdata(inputprefix=’yourprefix’, sumstatsfile=FALSE , depthfile=FALSE,geofile=NULL, nchroms=NULL)*

These are the function arguments:

inputprefix: prefix of RAW and BIM file (so without .raw and .bim extension)

nchroms: If your SNPs are mapped against a genome assembled to chromosomes, you could optionally provide the number of chromosomes.

samplefile: Name of optional input file (default is NULL) with population assignment. Input file should contain two tab-separated columns called ‘name’ and ‘pop’. The ‘name’ column contains the sample name (identical to names specified in ped-file) and the ‘pop’ column contains the population names (max 10 characters, no underscores or spaces). Optionally users can provide a third column called ‘pop2’, to specify subpopulation division within the main population division.

This file is not needed if the populations are defined in the first column of the ped-file (which is not the case if the ped-file has been converted from vcf.)

depthfile: If TRUE (default is FALSE), the function expects to find two files within the working directory named ‘out.idepth’ and ‘out.ldepth.mean’.

These files contain information about mean read depth per sample (idepth) and mean read depth per SNP (ldepth.mean), and can be generated using vcftools:

*vcftools --vcf inputfile.vcf --depth &*

*vcftools --vcf inputfile .vcf --site-mean-depth &*

This information will in subsequent steps be used to filter the dataset.

(Note that a vcf file created with ‘plink --recode vcf’ does not contain read depth information and therefore cannot be to used to calculate mean read depth.)

geofile: Name of optional input file (default is NULL) with geographical coordinates (longitude and latitude) of the samples. Input file should contain three tab-separated columns called ‘sample’, ‘longitude’ and ‘latitude’. Sample names in the geofile should be identical to sample names in PED-file. The coordinates should be in decimal degrees (e.g. New York: lat = 40.714, lon = -74.006). The information in the geofile will be used in subsequent steps to create geographical maps.

sumstatsfile: If TRUE (default is FALSE), the function expects to find within the working directory a file named ‘sumstats.tsv’, which is generated by the STACKS refmap pipeline. SambaR will use the information in the sumstats.tsv file to derive positions of the snps. You should include the sumstats.tsv file if you called your SNPs using the STACKS refmap pipeline, regardless of whether you created your PED/MAP files from a genepop file or vcf file. A genepop file does not contain information on the positions of your SNPs, and as result the first two columns of the MAP file wil contain zero’s only. A vcf file does contain information about the positions of your SNPs, but the vcf file created by STACKS shows the position of your SNPs in the STACKS loci rather than position in the reference genome. If you used the STACKS denovo pipeline or any other software to call your SNPs, you don’t need to provide a sumstats file, and therefore leave this flag to FALSE.

poolsfile: Name of optional input file (default is NULL) with the names of the sequencing library pool names of the samples. Input file should contain two tab-separated columns called ‘sample’ and ‘pool’. Sample names in the poolsfile should be identical to sample names in PED-file. The information in the poolsfile will in a subsequent step be used to create plots with show data quality per library pool.

colourvector: By default SambaR assigns, in alphabetical order of your population names, the following colours to your populations: darkblue (‘blue’), darkgreen, darkred, orange, purple (‘darkorchid4’), brown (‘tan4’), yellow, lightblue (‘deepskyblue’), lightgreen (‘greenyellow’), lightred (‘indianred1’) and darkgrey (‘gray20’). If you prefer other colours, or if you have more than 11 populations, you have to provided your own vector with colour names. The length of this vector should be equal or greater than the length of the numbers of populations defined in your input data. The vector should have the following syntax:

*c("blue","darkgreen","darkred","etc")*

For example, my inputfiles are called reindeer.raw and reindeer.bim. The data has been generated using STACKS refmap (so I have a sumstatsfile), and I subsequently used vcftools to calculate read depths stored in files called out.idepth and out.ldepth.mean. I also listed the geographical coordinates of the samples in a file called “geofile.txt”. I type:

*importdata(inputprefix=“reindeer”, sumstatsfile=TRUE , depthfile=TRUE,geofile=”geofile.txt”)*

Execution of the importdata() function will create a genlight object called ‘*mygenlight*’, and two dataframes, called *‘snps’* and *‘inds’*. More information about these objects in the next section.

***Trouble shooting***

If you forget to put double quotes around your inputprefix (e.g. inputprefix=reindeer instead of inputprefix=”reindeer”), you will get the error:

*Error in paste(inputprefix, "raw", sep = ".", collapse = NULL) :   
 object 'reindeer' not found*

As mentioned elsewhere in this manual, you might encounter the error:

*Error: unexpected input in "source(“"*

In that case your text editor has deformed the double quotes (from "" to “”). If so, either retype the double qoutes and run the command again, or copy paste via Notepad. This applies throughout this manual.

If by mistake you have set the sumstatsfile flag or depthfile flag to NULL rather than FALSE, you will get the error:

*Error in if (sumstatsfile) { : argument is of length zero*

*Error in if (depthfile) { : argument is of length zero*

If you generated the PED and MAP files using vcftools but did not provide input to the flag samplefile whilst running the importdata command, you will get the warning:

*WARNING: the number of populations in the 'popnames'-vector does not correspond with the number of populations in raw.file.*

*As a result, could not assign population names.*

*This will cause errors in subsequent functions.*

If there were hidden problems during file conversion (for example the file conversion from vcf to PED/MAP) stopped prematurely, you might encounter an error similar to:

*Error in read.PLINK(paste(prefix,"raw",sep=".",collapse=NULL));*

*some individuals do not have 86754 SNPs*

In that case likely something went wrong during the file conversions, resulting in one or more individuals not have the expected number of SNPs (as might be obvious from observing the PLINK). Redo the conversion and afterwards try to import the data again.

## 4.2 Import multiallelic data

SambaR’s functions to analyse multiallelic data sets are depreciated.

SambaR uses the ‘read.structure’-function of the Adegenet package to load multiallelic genotype data into R. This function expects your input file to be in the structure format with one row per individual. The first column should contain the sample names and the second column the population names. The remaining columns contain your genotype data, with two columns per locus. You can use PGDSpider to create this kind of structure file from almost any format.

Open the R session in which you loaded Sambar and use the setwd() command to navigate to the directory on your computer in which you stored your structure file:

*setwd("C:/Users/my/path/to/directory/”)*

Check whether your input structure file is present:

*list.files()*

If so, import your data. Say your structure file is called “mydata.stru”, you type:

*importmultidata(structurefile=”mydata.stru”)*

Execution of the importmultidata() function will create a genlight object called ‘***mygenlight***’, and two dataframes, called ***‘snps’*** and ***‘inds’***. More information about these objects in the next section. If everything works fine, all which should be displayed on the screen is the following message:

*Converting data from a STRUCTURE .stru file to a genind object...*

*Created two dataframes called 'snps' and 'inds', as well as a genlight object called 'mygenlight'.*

**Additional functions (still under construction):**

corrfrequencies()

allelefreqhisto()

meanr2() # simulate expected correlation fis and fst

## 4.3 Convert from genlight object

Rather than importing the data into R, users also have the option to create the SambaR data objects directly from a genlight object (i.e. an existing R object), using the function ‘genlight2sambar’. If your data is, for example, stored in a genlight object named ‘my\_gl’, the required SambaR objects can be created by running the command:

*genlight2sambar(genlight\_object=”my\_gl”)*

This function will generate two dataframes called snps and inds, a new genlight object called mygenlight, and a list object called mysambar. More information about these objects is provided in section 5 of this manual.

# 5. Observe your data

One of your challenges as a bioinformatician is to manage your input and output files. Because of the big variety of software and of filter settings, the number of files can quickly grow out of control. A main purpose of SambaR is to prevent this from happening, by storing all your data (input and most of the output) conveniently in no more than 3 R objects.

These 3 data objects are:

* A genlight object called ‘***mygenlight***’, containing the genotype data, with the number of columns equaling the number of loci, and the number of rows equaling the number of samples/individuals. 0 codes for 'homozygous major allele', 1 codes for 'heterozygous' and 2 codes for 'homozygous minor allele'. Missing data is denoted by ‘NA’.
* A dataframe called ‘***snps***’, containing locus specific information, with the number of rows equaling the number of loci.
* A dataframe called ‘***inds***’, containing sample specific information, with the number of rows equaling the number of individuals.

The ‘mygenlight’-object should not be manipulated. The user is free, in contrast, to add as many columns to the snps and inds dataframe as preferred (in addition to the columns generated by SambaR functions). If you are want to know more about the actual meaning of the columns of the snps and inds dataframe, you can find descriptions in section 10.1 and 10.2.

***Observe your data***

Because many SambaR functions add columns to the snps and/or inds dataframe, you might want to inspect your inds and snps dataframes regularly.

If you type on the R command line ‘mygenlight’, you will get a summary of the genotype data.

To observe a fraction of the actual data, type:

*as.matrix(mygenlight[1:10,1:10])*

To observe the first 5 lines of the snps file, type:

*head(snps,5)*

To observe the last 10 lines of the inds file, type:

*tail(inds,10)*

On some (not all!) platforms you can observe the snps file in R’s built-in data editor by typing:

*fix(snps)*

***Subselect your data***

To select a single column within the snps or inds dataframe you have to use the dollar sign. For example, if you want a summary of the identities of the minor alleles, you could type:

*table(snps$minor2)*

Alternatively, you could also type:

*table(snps[,”minor2”])*

The square brackets (‘[‘ and ‘]’) are for subselecting a dataset. In the part before the comma, you select rows. (If you don’t specify anything, like in in the example above, this is interpreted as: ‘select all’.) In the part after the comma, you select columns. If you want to select multiple columns and only the first 10 rows, it would look like this:

*snps[1:10,c("chr","name","pos","minor","major")]*

The findstructure() will export ordination plots, some of which will show sample numbers. If you want to know the name of a particular sample (for example sample number 51), you could type:

*inds[inds$nr==51,]*

or:

*inds$name[inds$nr==51]*

***mysambar***

The snps, inds and mygenlight objects are the three core objects used by SambaR, but in order to operate correctly it needs more objects. It needs to know for example the paths to the input and the output directories, and the font type to be used for the plots. All this information will be stored in a list object called ‘mysambar’. You can observe this list by typing:

*summary(mysambar)*

For normal use of SambaR, you don’t need to know about the existence of the ‘mysambar’ object. However, it is important to remember that you should NOT name any object ‘mysambar’, otherwise SambaR will stop working correctly.

# 6. Filter your data

To do quality control on your dataset, execute the following command:

*filterdata(indmiss=0.25,snpmiss=0.1,min\_mac=2,dohefilter=TRUE,min\_spacing=500,nchroms=NULL,TsTvfilter=NULL)*

Despite its name, the function does NOT remove data entries, not from the genlight object, nor from the inds or snps dataframes. In fact, it adds data. First, the function adds columns to the snps and inds dataframes with information about minor allele frequency, observed and expected heterozygosity, inbreeding coefficient, distance between snps, and number of missing datapoints. Based on this information the function will subsequently add columns with boolean vectors (i.e. FALSE/TRUE) to the snps and inds dataframes, indicating whether loci and individuals pass quality filters.

The filter settings you set now will be used for analyses. But if later on you decide you prefer other settings, you can quickly and easily rerun the analyses with different filter settings. For example, you could initially run the filterdata function with certain settings -- e.g *filterdata(indmiss=0.75, snpmiss=0.02)* -- prior to structure analyses, and afterwards rerun the function -- *filterdata(indmiss=0.05, snpmiss=0.15)* -- before doing genetic diversity analyses.

**Filter options**

The min\_mac flag of the filterdata function indicates how many copies of the minor allele a SNP should contain in order to be retained. In order to reduce the presence of genotype call errors, by default SambaR filters out SNPs which contain only 1 copy of the minor allele (i.e. min\_mac=2). If you want to retain all SNPs, set min\_mac to 0 or to NULL.

The dohefilter flag indicates whether you want to remove SNPs which are strongly out of Hardy-Weinberg-Equilibrium. These SNPs are highlighted in red in the ‘He\_vs\_maf’-plot which is generated by the filterdata function. Default is TRUE.

The snpdepthfilter flag indicates whether you want to remove SNPs which have unusual high read depth (if this information is provided by the user when importing the data). These SNPs are highlighted in red in the ‘Locusdepth’-plot which is generated by the filterdata function. Default is TRUE.

The min\_spacing flag serves to thin the data. The default setting of 500 indicates that SambaR will select maximum 1 SNP per 500 bp. For population structure, genetic distance and diversity analyses, SambaR functions will use the thinned dataset. This is to avoid issues related to linkage disequilibrium. For selection analyses, SambaR will use the non thinned dataset.

The TsTvfilter flag allows you to remove either transitions or transversions from subsequent analyses. Use the string ‘TsTvfilter==”Ts”’ to keep transitions only. Use the string ‘TsTvfilter==”Tv”’ to keep transversions only. Default is NULL.

If your SNPs are mapped against a genome assembled to chromosomes, you could optionally provide the number of chromosomes to the nchroms flag (default is NULL). If so, SambaR will generate additional plots showing the number of SNPs per chromosome, and try to determine the X-chromosome.

The most important flags of the filterdata function are the indmiss and snpmiss flags. These flags indicate the maximum allowed proportion of missing data. For example: if you set indmiss to 0.25 (default), all samples with more than 25 percent missing data will be ignored in subsequent analyses. Likewise, if you set snpmiss to 0.10 (default), all loci with more than 10 percent missing data (averaged over all samples which pass the indmiss threshold), will be ignored.

The default values for indmiss and snpmiss (0.25 and 0.10 respectively) are completely arbitrary and can be considered both conservative and liberal. You could argue a sample with 95 percent missing data is of low quality, but if the total dataset contains 50.000 snps, the sample still provides a goodly number of 2.500 non-missing datapoints.

Which values for indmiss and snpmiss should you choose? This depends on your research question. If you are mainly interested in structure analyses, you should probably opt for a strict snpmiss value (i.e. < 0.05, but preferably as low as possible). A relatively low number of high quality SNPs (with almost no missing data) might give you a more accurate picture of the true population structure than a high number of low quality SNPs. A number of 1000 high quality SNPs is often more than enough to infer population structuring, but this obviously does depend on your study system. Whereas the snpmiss value should be as low as possible, the indmiss value can be relatively high, as long you keep a sufficient number of SNPs. After running the importdata function, you will find in your SambaR directory a pdf-file called ‘Data\_quality’. Observe this file for guidance.

If you are mainly interested in genetic diversity analyses, you should keep in mind that samples with proportions of missing data likely had low coverage across the genome, which might have complicated the correct calling of heterozygosity. As a result the number of heterozygous sites in these samples might be biased downwards. A relatively small number of high quality individuals might give you therefore a more accurate estimate of genetic diversity within your populations than a high number of low quality samples. Observe the ‘He\_vs\_miss.pop’ plot for guidance. You want to set indmiss to a threshold such that for the retained dataset there is no relationship between sample heterozygosity and it’s proportion of missing data.

For selection and association analyses, for which your objective is to screen as many SNPs as possibly, you might get away with setting high value for snpmiss and indmiss. The presence of SNPs with high proportions of missing data might not cause major problems because selection analyses often work on a SNP by SNP basis and simply ignore missing data point (this might not be true for PCadapt though). On the other hand you could argue that the distribution of neutral SNPs (from which outlier SNPs need to stand out) might be affected by the presence of missing data, and that therefore some sort of filtering is desirable. To be on the safe side, you should probably run the selection analyses with various filter settings, and observe if the results are generally consistent.

**The output**

Once the execution of the filterdata function is finished, have a look in the Sambar output directory. You should find within the inputfiles directory PED and MAP files of the filtered datasets, both for the entire dataset (metapop) as for each population separately (if applicable), with two filter settings: filter1 and filter2 (more details below), and with genotypes either encoded in letters (A,C,G,T) or numbers (1,2,3,4).

Within the subdirectory ‘QC’ you will find a number of plots in up to 4 different formats (eps, pdf, png, wmf), so can select the format you prefer/need.

Also, columns have been added to the snps and inds dataframe. The overall filters are inds$filter and snps$filter (thinned dataset) and snps$filter2 (non thinned dataset). These columns determine whether samples and loci will or will not be used during subsequent analyses. If TRUE, the sample or locus will be included in the analysis. If FALSE, the sample or locus won’t be considered.

To avoid errors during the execution of subsequent functions, NEVER remove or rename the inds$filter, snps$filter and snps$filter2 columns.

**ADDITIONAL ANALYSIS: export files needed for additional analyses**

To calculate LD and sample relatedness/kinship with plink, as well as to calculate migration rates with Bayesass (see elsewhere for more info), you need to generate input files. To do so, execute the following function once the filterdata() function has finished:

*exportsambarfiles()*

**Exclude samples or populations from downstream analyses**

If you want to exclude particular samples from downstream analyses (say a sample called ‘sample5’, simply type:

*inds$filter<-inds$filter&inds$name!=”sample5”*

If want to exclude an entire population, say two populations called “pop2” and “pop5”, simply type (after the filterdata function finished):

*excludepop(do\_exclude=c("pop2",”pop5”))*

**Trouble shooting**

I mentioned this before, but it might be worth repeating:

If you decide to rerun the filterdata()-function, make sure that none of these plots are opened in a file viewer (like eps viewer or pdf viewer), otherwise this will result in an error similar to:

*Error in pdf("He\_vs\_maf.pdf", family = myfont, width = 10, height = 10) :*

*cannot open file 'He\_vs\_maf.pdf'*

If for whatever reason SambaR runs into an error whilst exporting a plot, the connection to this plot won’t be closed. This will prevent you from opening the plot (you will get an error saying that the file is in use by another device), and might also prevent R from exporting establishing new connections. Therefore, whenever you run into an error, it is good practice to, before running a SambaR function, execute the following command:

*dev.off()*

Keep on executing this command until you run into the following error message:

*Error in dev.off() : cannot shut down device 1 (the null device)*

**ADDITIONAL ANALYSIS: Plot PLINK relatedness (kinship) estimates**

SambaR has a built in function to estimate relatedness, but a more sophisticated method to calculated relatedness is implemented in PLINK. SambaR contains a function to plot sample relatedness measures (pi\_hat scores) generated by PLINK. For that purpose, you will find, if you have run the exportsambarfiles() function, within the inputfiles directory files called ‘metapop.filter.number.ped’, ‘metapop.filter.number.map’, ‘metapop.allinds.filter.number.ped’ and ‘metapop.allinds.filter.number.map’.

Navigate on the command line of your computer to this directory and execute these plink commands:

*plink --file metapop.allinds.filter.number –cow –-allow-extra-chr –genome -out plink.kin.allinds*

*plink --file metapop.filter.number –cow –-allow-extra-chr –genome -out plink.kin.retainedinds*

This will output to the SambaR input files directory a file called ‘plink.kin.allinds.genome’ and ‘plink.kin.retainedinds.genome’. In R, execute:

*plinkrelatedness(export=”pdf”)*

This function will export several plots to the Divergence subdirectory:

* Relatedness.plink.between.pdf
* Relatedness.plink.within.pdf
* Relatedness.plink.persample.pdf
* Relatedness.plink.matrix.pdf (with and without axes with sample numbers. The plots without axes have the legend included.)

**ADDITIONAL ANALYSIS: Plot GCTA relatedness (kinship) estimates**

SambaR contains a function to plot sample relatedness measures generated by GCTA.

Store the output of GTCA in SambaR’s inputfiles directory. Needed are 3 files, all with the same name, except for their extensions ‘grm.bin’, ‘grm.id’, and ‘grm.N.id’.

In R, execute:

*do\_gctamatrix(export="pdf",gctaprefix="yourprefix”)*

This function will export heatmaps to the Divergence subdirectory. It expects to find in the input files all individuals have been included in the analysis, also the samples which didn’t meet SambaR’s filter settings.

# 7. Analyse your data

Now you are ready to analyse your data. In this section I will discuss the SambaR functions which will help you to do so. Note that all subsequent functions expect to find:

* a genlight object called ‘mygenlight’
* a dataframe called ‘inds’, containing an inds$filter column
* and a dataframe called ‘snps’, containing a snps$filter’ and snps$filter2 column.

You created these objects in the previous steps, when importing and filtering the data.

## 7.1 Population structure

Likely the first thing you want to do is to find out about the structuring of your samples. In other words: you want to know whether your samples can be divided into distinct populations. This is especially interesting if you don’t have a priori information about population structure, but it is also useful to verify a priori expectations. To find out, execute the following command:

*findstructure(Kmax=6,add\_legend=TRUE,legend\_pos="right",legend\_cex=2,pop\_order=NULL)*

You can use the ‘Kmax’ flag to determine the number the maximum number of clusters considered in DAPC and admixture analyses. You can use the ‘pop\_order’ flag to determine the order of your populations in the admixture plots. For example: if your populations are named ‘America’, ‘Asia’, ’Europe’, you can specify an alternative order from left to right by typing:

*findstructure(Kmax=6,pop\_order=c(‘Europe’,’Asia’,’America’)*

This function will produce a lot of rubbish on the screen, and in the meantime perform many analyses, including PCoA, PCA, MDS, DAPC, Bayesian Population Assignment tests, and admixture analyses. If the columns longitude and latitude are present in the inds dataframe, it will also produce geographical maps.

Samples will be coloured according to your population assignments. If you have defined one population only, all samples will have the same colour. Some of the PCoA, PCA and MDS plots will show sample numbers. These plots are not meant for publication (i.e small font size), but are for you to figure out which samples clusters together.

**Different symbol types based on sample traits**

You can optionally create PCoA plots with multiple symbol types based on a certain criterium. For example, to use different symbol types for male and females, first create a txt file with columns named ‘name’ and ‘sex’, store this file in the directory with stores your PED and MAP files, and next use the addsample function to insert sex information into the inds dataframe:

*addsampleinfo(samplefile="sample\_info.txt",filteronly=FALSE)*

This will add a column called ‘sex’ to the inds dataframe.

Next execute:

*ape\_pcoa(method="hamming",symbolkey="sex",legendpos="topleft",legendcex=2,export=”pdf”)*

**Change the size and position of the legend in ordination plots**

If the legends are overlapping with the samples, run the findstructure() function again without legends, or with legends positioned at different plot regions:

*findstructure(add\_legend=FALSE)*

*findstructure(add\_legend=TRUE,legend\_pos=”topleft”)*

The legend\_pos flag accepts as input 9 different strings: “topleft”, “top”, “topright”, “left”, “center”, “right”, “bottomleft”, “bottom”, or “bottomright”.

You can also edit the size of the legend labels and the the size of the dots:

*findstructure(add\_legend=TRUE,legend\_pos=”topleft”,legend\_cex=1.5,symbol\_size=1.5)*

Legend\_cex and symbol\_size are by default set to 2.

***Trouble shooting***

Depending on the size of your data, the settings of your computer, and on the last time you closed the computer, you might run into the following error:

*Creating Nei's genetic distance matrix... Error: cannot allocate vector of size*

In that case you could try restarting the computer or omit the nei’s D calculations:

*findstructure(do\_nei=FALSE)*

Occasionally, you might run into the following error:

*Creating pcoa plot based on nucleotide diversity using Sambar functions and Ape...*

*Error in array(STATS, dims[perm]) : 'dims' cannot be of length 0*

This error occurs when running Adegenet’s pcoa function on a matrix on pairwise sequence dissimilarity estimates. I don’t know how to solve this error other than to omit this particular analysis by typing:

*findstructure(do\_pi=FALSE)*

***Trouble shooting***

Occasionally LEA might throw the error:

*Error in snmf("LEAinput.geno", K = ndemes, alpha = 100, project = "new") :*

*internal error in trio library*

Simply rerunning it (multiple times) might solve the problem, or else try lower the K-value:

*findstructure(onlyLEA=TRUE,Kmax=5)*

**Reassign samples to populations and rerun analyses**

Based on the output of the findstructure()-function, you might want to reassign your individuals to populations, and redo some of the calculations you did before (like minor allele frequency and heterozygosity per population). This is true if you didn’t have a priori expectations about population stratification, but it might also be the case if your a priori expectations turned out to be incorrect.

A quick way to reassign samples is to create a (new) samplefile as explained elsewhere in this manual and rerun the importdata() and filterdata() function. Or try function replacepop():

*replacepop(samplefile=“filename.txt”)*

You might occasionally find that a few individuals stand out, and that all other individuals are clumped together. If so, you could try to rerun the findstructure() function, this time excluding the individuals which stand out. To do for example the analysis again without individuals numbered 6, 20, and 43, type first:

*inds$filter[c(6,20,43),] <- FALSE*

**Change the size of the piecharts in the geographical maps**

If you want to change the size of the piecharts in the geographical maps, you can recreate these plots by running the function:

*create\_sambarmaps(K\_max=6,radius\_ratio=50)*

The radius\_ratio flag determines the size of the piecharts. It defines the radius of the piecharts relative to either the longitude or latitude range (smallest of either). For example: if your longitude values range from 10 to 20, your latitude values from 25 to 30, and if radius\_ratio is set to 50, the piecharts will be (30-25)/50 = 0.1 latitudes.

**ADDITIONAL ANALYSIS: Plot Admixture output (or any other structure-like output)**

On Linux install Admixture in any directory you like:

*wget http://software.genetics.ucla.edu/admixture/binaries/admixture\_linux-1.3.0.tar.gz*

*tar –zxvf admixture\_linux-1.3.0.tar.gz*

Run in R the function:

*exportsambarfiles()*

This will create several files in SambaR’s inputfiles directory, including files called ‘metapop.filter.number.ped’ and ‘metapop.filter.number.map’. Copy these file into the newly created ‘admixture-1.3.0’ folder, and execute (using 4 threads, as defined by ‘j4’):

*./admixture –cv metapop.filter.number.ped 2 –j4*

You can run the software several times, with different numbers of K. For example, if you want to run to software for K=3 instead of K=2, type:

*./admixture –cv metapop.filter.number.ped 3 –j4*

All structure like analyses output a matrix with number of columns equalling K and rowsums equalling 1. In the case of Admixture this matrix is stored within a file which has the extension Q. Save the matrices in separates files in SambaR’s inputfiles directory, ending on the suffix ‘Kn.qmatrix.txt’, with *n* indicating the value of K (i.e. number of columns). For example, if you did the analyses for K2, K3 and K4, you should have three files, ending on the names ‘K2.qmatrix.txt’, ‘K3.qmatrix.txt’ and ‘K4.qmatrix.txt’. Get rid of headers and of the column with sample names, but you need to make sure that the rows are ordered on sample name (as in the inds$name column), which is the case for the ‘metapop.filter.number.ped’ file.

To plot all matrices together in one plot, execute:

*plotstructure(export=”pdf”,addindnr=TRUE,order\_on\_longitude=FALSE)*

## 7.2 Population differentiation

Now that you have identified the populations within your dataset, the next step is to calculate the genetic distances between those populations. Which populations are more similar, and which are more divergent?

To find out, execute the following command:

*calcdistance(nchroms=NULL)*

Note: it only makes sense to run this command if you have defined more than 1 population. If all samples are assigned to the same population, Sambar will not perform any calculations.

This function will calculate several genetic distance measures (i.e. Fst and Nei’s genetic D) for all pairwise population comparisons. I will also calculate D-statistics.

If your SNPs are mapped against a genome assembled to chromosomes, you could optionally provide the number of chromosomes to the nchroms flag (default is NULL). If so, SambaR will generate karyotype plots with genetic distance measures on a sliding window basis.

## 7.3 Genetic diversity

You might also want to know how much genetic variation each population contains.

To find out, you can execute the following command:

c*alcdiversity()*

To get the most out of this function, you need to provide a value to the nrsites argument (default is NULL). This value is the combined length of the sequences from which your SNP dataset is obtained, including the sequences which did not contain SNPs.

If you generated your SNP dataset with the software STACKS, this value can be obtained from the populations.log file. The end of this file contains a line stating: ‘Kept ……. loci out,’. (For older versions of STACKS, the line reads: ‘Of these, ……. loci/stacks passed the filters.) Multiply this number with the length (in basepair) of your trimmed reads (e.g. 110 bp) and you have a rough estimate of the number of retained sequenced sites.

If you generated your data using whole genome resequencing, you can obtain an estimate of nrsites by generating a vcf file which contains both polymorphic and monomorphic sites. Instructions are provided in the text box called ‘How to obtain an estimate of nrsites’ (see below).

If for example the combined length of your loci is 9350000 base pairs, the command would be:

c*alcdiversity(nrsites=* *9350000)*

If you provide a value to the nrsites flag, the function will calculate estimates of genome wide heterozygosity, nucleotide diversity and Watterson’s theta and Tajima’s D. It will also generated 1 and 2 dimensional folded site frequency spectra (SFS):

**How to obtain an estimate of ‘nrsites’**

To obtain an estimate of nrsites, there are two things to keep in mind when calling SNPs:

1. Make sure that the output file (e.g. vcf file) contains both polymorphic and monomorphic sites, rather than monomorphic sites only. If you use for example the bcftools mpileup/call pipeline, do NOT include the -v option when running the bcftools call command, because otherwise monomorphic sites are not included in the VCF output file.
2. Wait with extracting (biallelic) SNPs until you finished filtering the vcf file. This way you can easily count the total number of sequenced and retained sites, which is the number of sites prior to extracting biallelic SNPs.

Here are instructions for using the bcftools mpileup/call pipeline:

STEP 1. Calculate the genotype score for each position (monomorphic and polymorphic):

*bcftools mpileup -A -ugf reference.fa -b bamfiles.txt > allsites.bcf  
bcftools call -m allsites.bcf > allsites.vcf*

# Note: don't include the -v option, otherwise monomorphic sites won't be included.

STEP 2. Filter the data:

*bcftools view -O z --include 'DP>=8' allsites.vcf > allsites.mindepth8.vcf.gz*

*vcftools --vcf allsites.mindepth8.vcf.gz --remove-indels --minQ 30 --max-missing 0.95 --recode --recode-INFO-all --out allsites.filtered &*

STEP 3. Extract biallelic SNPs:

*vcftools --vcf allsites.filtered.recode.vcf --min-alleles 2 --max-alleles 2 --recode --recode-INFO-all --out SNPsonly.filtered &*

The estimate of nrsites, the total number of sequenced sites in the retained dataset (from which the SNPs are extracted), is simply:

*grep -v '#' allsites.filtered.recode.vcf | wc -l*

**ADDITIONAL ANALYSIS: Linkage disequilibrium**

SambaR contains a function to plot LD output generated by the software PLINK. For that purpose, you will find in the inputfiles directory (if you executed the exportfiles() function), PED and MAP files ending on ‘filter2.number.map’ and ‘filter2.number.ped’. (Note: for this analysis you want to use ‘filter2’, not ‘filter’.

To calculate linkage disequilibrium, navigate on the command line of your computer (so not in R) to the Sambar\_output directory, and execute for each population the following command:

*plink --noweb --cow --allow-extra-chr --file yourpop.filter2.number --r2 --ld-window-kb 250 --ld-window-r2 0 –out yourpop*

Be sure to replace ‘yourpop’ with the name of one of your populations, used by Sambar. For example: if your file is called Busen, it becomes ‘Busen.filter2.number’.

This command will create three files, ending on ‘ld’, ‘log’, and ‘nosex’. The ld file will contain LD values for each pairwise combination of snps occuring on the same contig or chromosome within a distance of 1.0 Mb. To plot these estimates, execute on the R command line:

*LD\_plot(export=”pdf”,* *xrange=c(0,1000000),stepsize=100000)*

This function outputs a file called ‘LD.boxplot.100K.pdf’ to the QC directory. This file shows LD values for each of your populations. You can export the plot in different formats by setting the export flag to ‘eps’, ‘png’, or ‘wmf’. If wanted you can edit the range and the stepsize.

If your reference genome is assembled up to chromosomes (rather than contigs or scaffolds), you can also execute:

*LDperchrom()*

This function will export a table with mean LD estimates per chromosome.

## 7.4 Selection analyses

You might be interested in the question if and which loci are (possibly) under diversifying or stabilizing selection. To find out, SambaR executes 4 interpopulation selection scans: Fsthet, GWDS, OutFlank, and PCadapt. (GWDS can detect diversifying selection only.) In addition, SambaR exports files which can be used as input for Bayescan and Bayenv, and optionally provides sliding window Fst and Tajima’s D analyses.

There are three main options:

* ***Analysis for the metapopulation***

The default option is to run selection analyses for the metapopulation, i.e. considering all populations at the same time. This only works for Fsthet, OutFlank and PCadapt (and optionally Bayescan), as GWDS works for pairwise comparisons only.

* ***Analyses for all population pairs***

The second option is to run selection analyses for all pairwise population comparisons. Depending on the number of populations in your dataset, this can result in a higher number of analyses.

* ***Analyses for ‘pheno1 vs pheno2’***

The third option is to divide your populations in two groups, and execute selection analyses only for pairs of populations which don’t belong to the same group, as well as analysis of group A vs group B. The division can be based on either a phenotypic trait or on geographical occurence. To group your populations, you have to add to your inds dataframe a column called ‘inds$type’ which contains TRUE/FALSE values. For example: if your data contains two marine populations (pop1 and pop2) and two freshwater populations, the individuals of the marine populations would get the value ‘TRUE’ and the remaining individuals would the value ‘FALSE’. One way to create this vector is:

*inds$type<-ifelse(inds$pop=="pop1"|inds$pop=="pop2",TRUE,FALSE)*

If you want to run selection analyses only for group A vs group B, leave the flag onlypooled to TRUE. If you want to run selection analyses for all pairwise comparisons between group A and group B populations, set the flag onlypooled to FALSE.

To execute each method, type:

*selectionanalyses(do\_meta=TRUE* ,*export=”pdf”*,*do\_fsthet=FALSE,my\_correction=NULL)*

*selectionanalyses(do\_meta=FALSE,do\_pairwise=TRUE,export=”pdf”* *do\_fsthet=FALSE)*

*selectionanalyses(do\_meta=FALSE,do\_pheno=TRUE,onlypooled=TRUE,export=”pdf”,phenolabels=c(“high”,”low”),* *do\_thin=FALSE,gwdsbinsize=200000,my\_correction=NULL)*

The selection scan ‘Fsthet’ takes a relatively long time to run (like 30 minutes for a dataset of 60 individuals and 50,000 loci), and is therefore excluded from the analyses by default. If you want to include this scan, set the do\_fsthet flag to TRUE. This analysis requires you to have a genepop file of your original input data in the input data directory (i.e. same directory as where RAW/BIM files are stored.)

The my\_correction flag can be used to define the desired multiple test correction method. Values can be NULL (default), ‘bonferroni’, ‘holm’, and ‘BH’, the latter denoting Benjamini-Hochberg. If NULL, GWDS will be run using the Bonferroni correction, PCadapt will be run using the Holm correction holm, and OutFLANK will be run using q-values.

The do\_thin flag applies to the GWDS test, and indicates whether you want to infer the neutral distribution from a thinned dataset. The default is FALSE (applicable to RADseq SNPs datasets), but if it turns out your dataset contains many SNPs belonging to the same regions under selection, you are likely to get better results (i.e. less conservative threshold for defining outliers) by setting this flag to TRUE. If set to TRUE, SambaR will use one SNP per genomic region to infer the neutral distribution. The size of these genomic regions can be adjusted with the gwdsbinsize flag (default is 200000 bp).

Note: the option ‘do\_thin’ should be set to TRUE only if working with dense SNP datasets. When setting the flag ‘do\_thin’ to TRUE for datasets with relatively low number of SNPs (e.g. RADseq SNP datasets), GWDS will return many false positives.

**The output**

Identified outliers are potentially under either positive/diversifying selection or balancing selection. (GWDS is the exception, as it can detect diversifying selection only). You can infer the type of selection from the piecharts and the WC\_Fst.outliers plots. Note that outliers are just that: outliers. They are either loci under selection or, not unlikely, false positives. Have a look at the allele frequencies of the outlier SNPs and consider whether the allele frequencies are similar or dissimilar enough between/among populations (compared to the differences in other SNPs) to justify the inference of selection. See the section ‘Simulate your data’ within this manual for a potential way to obtain power and specificity estimates of the selection scans GWDS, PCadapt and OutFLANK in the context of your study system.

SambaR’s GWDS (Genome Wide Differentiation Scan) implements the Fisher exact fisher test to search for correlations between (groups of) populations and their minor allele frequencies (using R’s build in fisher.test function). Afterwards it fits an exponential curve to the histogram of obtained p-values, assuming the distribution is exponential (using R’s build in rexp function). Using this exponential curve, a Bonferroni corrected threshold value is deduced (using R’s build in qexp function).

SambaR outputs Manhattan plots of the negative log of the p values based on Fisher exact tests. It will also add three columns to the snps dataframe: snps$rfisherp, and snps$rfisherlogp, and snps$rfisherout. SambaR will also output histograms of the p values, and the fitted curve. Users should check if the fit is good. If not, the outcome of the GWDS test might be unreliable.

***Trouble shooting***

You might run into the one of the following errors:

*Calculating FSTs, may take a few minutes...*

*Error in if (s2 == 0) { : missing value where TRUE/FALSE needed*

*All loci with Fst above the upper (righthand) trim point were marked as outliers. Re-run with smaller RightTrimFraction or smaller qthreshold.*

*Error: $ operator is invalid for atomic vectors*

*Error in quantile.default(pi0, prob = 0.1) :*

*missing values and NaN's not allowed if 'na.rm' is FALSE*

*Error in optim(NumberOfSamples, localNLLAllData, lower = 2, method = "L-BFGS-B") :*

*L-BFGS-B needs finite values of 'fn'*

*Error in Sample\_Mat[, 1] : subscript out of bounds*

These errors are all related to OutFlank and unfortunately I don’t know how to fix them. The only solution I can offer at the moment is to exclude OutFlank from the analyses. For example:

*selectionanalyses(do\_meta=TRUE,* *do\_outflank=FALSE)*

*selectionanalyses(do\_meta=FALSE,do\_pairwise=TRUE,* *do\_outflank=FALSE)*

**ADDITIONAL ANALYSIS: Bayescan**

The selection scan Bayescan runs on the Unix command line, can therefore not be invoked by SambaR. However, SambaR does accept Bayescan output. To create input files for Bayescan, first create the inds$type column as explained above, and then run:

*createbayescaninput(allpairwise=FALSE)*

To execute PGDspider and Bayescan, you first have to convert the PED and MAP files to Bayescan input format (called ‘geste’). To do so, execute on the command line (assuming java have and PGDSpider have been installed):

*java -jar /path/to/PGDSpider2-cli.jar -inputfile pheno.filter2.letter.ped –outputfile pheno.filter2.geste -spid /path/to/PED2Bayescan.spid &*

The PED2Bayescan.spid file is included in your SambaR download.

If you receive the message ‘Please specify an input file’, then PGDSpider is not able to find your input file, maybe because you misspelled it or did not provide the right path.

No need to worry if you receive the error message ‘simdata.map not found!’.

Now you are ready to run Bayescan. First create an output directory:

*mkdir bayescan\_pheno*

Then run Bayescan:

*/path/to/bayescan\_2.1 pheno.filter2.geste -od bayescan\_pheno -threads 10 &*

If Bayescan can not find the output directory, it will give the somewhat confusing error ‘Could not open file’.

To subsequently incorporate Bayescan output in Sambar plots, transfer bayescan output files ending on either ‘baye\_fst.txt’ or ‘g\_\_fst.txt’ to the SambaR inputfiles directory, rename them with ‘pop1\_pop2.bayescanout.fst’ (or ‘pheno.bayescanout.fst’), and execute in R:

*selectionanalyses(do\_meta=FALSE,do\_pheno=TRUE,add\_bayescan=TRUE)*

**ADDITIONAL ANALYSIS: Sliding window Tajima’s D**

*wintajd(my\_chrom=25,winsize=1000000,winstep=200000)*

This function will export (in pdf format) graphs with sliding window Tajima’s D estimates. It will also produce a dataframe called wintaj.

**ADDITIONAL ANALYSIS: Find and plot genes close to outlier SNPs**

If you used reference mapping rather than denovo mapping, Sambar will have exported to the selection directory txt.files in BED format. These files contain lists with outlier SNPs, and can be used to find nearby genes using the software BEDTOOLS2 . To do so, you need the have the annotation file (in gff format) of the genome which you used as reference.

On Unix command line execute the following command to convert this annotation file from gff format to BED format:

*cut -f1,4,5,9 reindeer\_coding\_gene\_annotations.gff > genes.bed*

Sort both BED files:

*bedtools2/bin/sortBed -i reindeergenes.bed > genes.sorted.bed*

*bedtools2/bin/sortBed -i outliers.bed > outliers.sorted.bed*

To find the 10 closest features (including overlapping features), and to report the distance between SNP and feature (-D a flag), execute:

*bedtools2/bin/closestBed -a outliers.sorted.bed -b genes.sorted.bed -D a -k 10 > putative.txt*

To subsequently select features which are within 200kb distance from an outlier SNP:

*awk '$9 <= 200000' putative.txt | cut -f1-8 | uniq -f7 | cut -f1,6,7,8 > putative. 200kb.bed*

It might be that the gene names (4th column of BED file) are uninformative, and that you have to perform a blast with the gene sequence to find the actual gene name.

To plot the positions of the outlier SNPs and the adjacent genes, copypaste the BED-file to the selection subdirectory and execute:

*multiplotscaffold(my\_bed=”putativegenes.within200kb.genenames.bed",background\_pop=NULL,doexport=TRUE,x\_range=NULL,y\_loc=c(0.4,0.65))*

This will export pdf files of all contigs/chromosomes containing outlier SNPs to the selection subdirectory. With the background\_pop flag you can determine which population is shown on the background. With the x\_range flag you can determine the region of the contig/chromosome you want to display. With the y\_loc flag you can edit the locations of the gene names.

## 7.5 Association analyses

The selectionanalyses() function allows you to detect SNPs which have allele frequencies which differ strongly between your populations. Alternatively, you might want to find out if your dataset contains SNPs which have allele frequencies which differ strongly between individuals within populations, for example between males and females, or between healthy and affected individuals. This type of analysis is known as genome wide association scan (GWAS).

To do GWAS with SambaR, first use the addsampleinfo() function to add additional sample specific information to the inds dataframe.

*addsampleinfo(samplefile="sample\_info.txt",filteronly=TRUE)*

Assuming this will have added a column called ‘sex’ to the inds dataframe, next execute:

*assocfisher(pheno="sex",export="pdf",mylabels=c("male","female"))*

This will run GWDS (as explained in the previous section) and output to the selection directory a ‘WC\_Fst.outliers.pdf’ file which highlights outlier SNPs (if present) in blue.

**ADDITIONAL ANALYSIS: RDA**

Create an input file with geographical coordinates, sample names (called ‘individual’), and environmental variables, and store this txt file (for example under the name “evofile.txt”) in the directory which also contains the PED and MAP input files. Next execute:

*runRDA(export="pdf",legendpos="topleft",envfile=”evofile.txt",doall=FALSE)*

## 7.6 Population demography (ADDITIONAL ANALYSIS)

Genetic diversity estimates can be used to estimate current and historic effective population sizes. One program you can use to do so is the Stairway\_plot executable (<https://sites.google.com/site/jpopgen/stairway-plot>). To get this program up and running simply download the latest version and unzip the file. Hamilton users can copy the program from: */ddn/data/fjsq43/Programs/Populationdemography/stairwayplot/stairway\_plot\_v2*

You don’t need your genepop data to run stairway\_plot analyses. The program takes all the information it needs from the site frequency spectrum (SFS) vector, which SambaR generated when executing the calcdiversity() function. You need to input the information in this vector into an input file, as I will explain below.

If you navigate within the folder ‘stairway\_plot\_v2’ you will find two files called ‘two-epoch.blueprint’ and ‘two-epoch\_fold.blueprint’. Since you created a folded SFS vector (i.e. a ‘minor allele frequency SFS’ rather than a ‘derived SFS’, since you don’t know whether an allele is ancestral or derived), you have to use the second one.

For each of your populations, copy, edit and save the ‘two-epoch\_fold.blueprint’ file. See you for now you save it under the name ‘mypop.blueprint’. The lines you have to edit are indicated in bold below. For L (total number of sequenced sites) you should use the same number as the input to the nrsites argument for the calcdiversity() function. (For more information, see section 7.3 of this manual.) On the SFS line simply copy paste the line in the file SFSvector.yourpop.txt (‘Diversity’ subdirectory).

#example blueprint file

#input setting

**popid: yourpop # id of the population (no white space)**

**nseq: 42 # number of sequences (length of SFSvector x 2)**

**L: 9350000 # total number of sites**

whether\_folded: true # whether the SFS is folded (true or false)

**SFS: 1946 1032 951 901 etc # SFS vector, generated with calcdiversity() function**

smallest\_size\_of\_SFS\_bin\_used\_for\_estimation: 1 # default is 1; to ignore singletons, change to 2

**largest\_size\_of\_SFS\_bin\_used\_for\_estimation: 21 # default is nseq/2 for folded SFS**

pct\_training: 0.67 # percentage of sites for training

**nrand: 10 20 30 40 # (nseq-2)/4, (nseq-2)/2, (nseq-2)\*3/4, nseq-2, space separated**

**project\_dir: yourpop # project directory name**

stairway\_plot\_dir: stairway\_plot\_es # directory to the stairway plot files

ninput: 200 # number of input files to be created for each estimation

#output setting

**mu: 2.5e-8 # assumed mutation rate per site per generation**

**year\_per\_generation: 20 # assumed generation time (in years)**

#plot setting

**plot\_title: yourpop # title of the plot**

xrange: 0.1,10000 # Time (1k year) range; format: xmin,xmax; "0,0" for default

yrange: 0,0 # Ne (1k individual) range; format: xmin,xmax; "0,0" for default

xspacing: 2 # X axis spacing

yspacing: 2 # Y axis spacing

fontsize: 12 # Font size

Now you are ready to run the analyses. The following instructions are for Linux.

First make sure Java is added to the environment (Hamilton users, type: *module add java*).

To run the program all you have to do is to execute for all your populations two commands.

First execute:

*java -cp stairway\_plot\_es Stairbuilder mypop.blueprint &*

This command will create within seconds new directories and a bash script.

Next execute the bash script:

*bash mypop.blueprint.sh & # note: don’t forget the extension (.sh)*

This command will initiate the actual calculation, which takes typically several hours or days to complete, depending on the total number of sites. It prints zero’s on the screen whilst running.

***Plot results***

The Stairway\_plot executable automatically generates within the output directory plots (both in png and pdf format) which can be readibly viewed. To generated a multitile plot with shows the output of all populations combined, with colour settings which agree with the other plots generated by the SambaR-package, SambaR comes with a function which recreates the plots outputted by the Stairway\_plot executable.

To make use of this function, transfer the files ending on ‘.final.summary’ to the Demography directory. You should have as many of those files as populations (listed in the populations vector), and the prefices of the files should correspond with the names listed in the populations vector. For example, if according to your populations vector you have three populations, called ‘Busen’, ‘Barff’ and ‘Norway’, SambaR expects to find within the Demography directory three files, called ‘Busen.final.summary’, ‘Barff.final.summary’ and ‘Norway.final.summary’.

Now execute the function:

*run\_plotstairway(mu\_rate="2.5\*10^-8",Gtime="20",x\_range=c(100,17500),exporttype="pdf")*

Set u\_rate and gen\_time to the correct values (the values used whilst generating the stairwayplot output files). This information is used for labelling the plot. Set x\_range for values greater than 0. The values for x\_range should be in years (i.e. not in ky or My).

This function will export multiple files called ‘stairwayplot.pdf’ to the ‘Demography’ subdirectory. The plot can also be exported in eps format by setting the export flag to eps.

**Background information about the site frequency spectrum (SFS)**

A SFS is a histogram showing the frequencies/numbers of SNPs binned based on their number of minor allele copies. Say for example that we have the following SFS-vector:

*3678 560 400 322 360 78 20 0 0*

This SFS-vector tells us that for a given (meta)population the dataset contains in total 5418 SNPs. Of those, 3678 SNPs have 1 copy of the minor allele, 560 have 2 copies of the minor allele, 400 sites have 3 copies of the minor allele, 322 have 4 copies of the minor allele, etc.

It also tells us that the dataset consists of 9 individuals, because assuming diploidy and assuming the data contains biallelic SNPs only (i.e. no SNPs with more than 2 different alleles) the total number of allele copies per SNP (minor and major allele combined) is twice the number of individuals. As the minor allele is by definition the less abundant allele, the minor allele can be represented by maximum half the number of total allele copies, which for diploid individuals equals the number of individuals. Therefore, the length of the SFS-vector equals the number of individuals, which in the example above is 9. In the example above there are no SNPs with 8 or 9 copies of the minor allele, and therefore the vector ends with 2 zero’s.

***How does SambaR generate SFS-vectors?***

For each population, SambaR computes for each SNP the number of minor allele copies using the glSum function of the adegenet package.

The minor allele is defined based on the total dataset, so it could be that within certain populations, the minor allele has more copies than the major allele. If that is the case for a SNP, SambaR subtracts for this particular SNP the number of minor allele copies from the total number of observed allele copies for this SNP (ignoring missing data point). This latter number is computed with the glNA-function of the adegenet package.

The output of this first step is saved in columns in the snps dataframe. The name of these columns start with mac (for minor allele count), followed by an underscore and the name of the population. In the second step SambaR computes the number of occurrences of each value in this column. You could roughly do so yourself by typing on the command line:

*table(snps$mac\_popname) # replace ‘popname’ with the name of one of your populations*

By default, Sambar includes in the calculation all SNPs, also the ones which did not pass filter settings. In contrast, it does not include individuals which did not pass the filter settings. The SFS vector generated by SambaR does not include sites with zero copies of the minor allele. Therefore, the length of the SFS-vector should equal the number of retained individuals.

**ADDITIONAL ANALYSIS: Generating SFS vectors with ANGSD instead of with SambaR**

Another way to acquire SFS vectors is by using the software ANGSD. You can generate this SFS vector, starting from your bam files, in just two commands. For each of your populations run the following command:

*/path/to/angsd/angsd –bam listofbamfiles.txt –doSaf 1 –anc /path/to/referencegenome.fa –GL 1 –out yourpopname –fold 1 –minMapQ 20 –minInd 20 –nThreads 4 &*

Afterwards run:

*/path/to/angsd/misc/realSFS yourpop.saf.idx -p 4 > yourpop.sfs*

Note that, in contrast to the SFS vectors generated by SambaR, the SFS vector of ANGSD includes the number of non-segregating sites. Therefore you will notice that the first value of the SFS vector is much higher than the other values. You will also notice that the length of the vector is the number of individuals (as defined in the listofbamfiles.txt) plus 1.

When preparing the blueprint file of the Stairway plot analyses, the L value should be the sum of the SFS vector. On the line of the SFS vector, you should insert the SFS vector except without the first value.

Because ANGSD uses a different method to estimate the SFS vector (a likelihood method), the value might not be rounded. To make it more realistic, you might want to round the values.

**ADDITIONAL ANALYSIS: Circosplot with Bayesass migration rates**

SambaR contains functions to create input files for Bayesass3-SNPs, and to subsequently create a circos plot of the migration rates calculated by the software Bayesass3-SNPs. Execute:

*exportsambarfiles()*

Afterwards, you wil find in the inputfiles directory a file called ‘Bayesassinput.immanc.txt’.

Download Bayesass (from: https://github.com/stevemussmann/BayesAss3-SNPs), unzip on Unix command line (type: unzip BayesAss3-SNPs-master.zip), copy the Bayesassinput.immanc.txt file to the newly created Bayesass directory, and execute:

*BA3-SNPS-Ubuntu64 --file Bayesassinput.immanc.txt --loci 15000 -s 10 -i 1000000 -b 100000*

This will run Bayesass with the defaults settings of 1 million iterations, a burn-in of 100000 generations, a seed of 10, and with delta (A, F and M) values of 0.10000000000000001, and for 15000 loci. Set the number of loci equal to:

*nrow(snps[snps$filter,])*

After completion of the run, which can take hours, you will find a file called BA3out.txt. In it you will find a matrix following the line Migration Rates which looks similar to:

*m[0][0]: 0.8973(0.0348) m[0][1]: 0.0066(0.0028) m[0][2]: 0.0554(0.0209)*

*m[1][0]: 0.0138(0.0086) m[1][1]: 0.8262(0.0610) m[1][2]: 0.1110(0.0575)*

*m[2][0]: 0.0016(0.0012) m[2][1]: 0.0043(0.0031) m[2][2]: 0.9895(0.0036)*

(If you can’t find it, type on command line: grep –A10 -B4 'Migration Rates' BA3out.txt)

This matrix shows proportion of migrants per generation (proportion of population size) – the direction being from column to row (e.g: 0.0514 migrants from pop2 to pop1). Copy paste the matrix into a new file in SambaR’s inputfiles directory (e.g. name it ‘bayesassmatrix.txt’). Add to the beginning of each row the name of the specific population, followed by a space or tab.

Now, to visualize migration rates in a circos plot execute the following SambaR function:

*plotmigration("bayesassmatrix.txt",export=”pdf”,addlabels=TRUE)*

The circosplot will be exported to the Divergence subdirectory.

***Potential errors***

If you input file doesn’t end with a white line, you might get the error:

*In read.table(myinputmatrix, stringsAsFactors = FALSE) :*

*incomplete final line found by readTableHeader on ‘’*

If so, add a white line to the end of input file.

# 8. Manage your data

***Export your data***

Although SambaR comes with built-in functions for population genetic analyses, it obviously can not do everything for you. For that reason, the SambaR package contains a function which allows you to export your filtered data in PED and MAP format, which can be used as input for programs which don’t run in R.

The function exports the combined dataset as well as all populations separately, meaning you can easily perform calculations on the entire dataset as well as on populations separately.

To export your data to the SambaR directory, execute the following command:

*exportdata()*

***Trouble shooting***

If the population names listed in the mysambar$populations vector do not correspond with the names in the inds$pop2 column, you will get the error:

*Error in x@gen[[1]] : subscript out of bounds*

***Back up your data***

You might find yourself working at multiple projects at the same time. An easy way to switch between datasets is by saving your genlight, inds, snps and population objects in a sambar list object. It might be wise to back up your data anyway. If you make a mistake and somehow delete your data, it allows you to easily retrieve it (see section ‘reload your data’).

To back up your data, for example under the name ‘myproject\_1’, type:

*backupdata(“myproject\_1”)*

The data, including the paths to directories and font type vector, is stored in an list object. These types of objects are a bit more difficult to work with than dataframes, but still not that hard. Some commands to observe the backup of your data are:

*myproject\_1$genlight*

*summary(myproject\_1)*

***Reload your data***

To reload your data to the populations, snps, inds and mygenlight objects (as well as paths to directories and font type vector), type:

*getdata(“myproject\_1”)*

Be careful! This will overwrite the existing populations, snps, inds and mygenlight objects, so be sure to back up those files first if you haven’t do so yet.

Reloading a dataset is obviously only possible if you previously backed up this dataset using the the backupdata()-function. Be sure to use the same prefix as when running the backupdata()-function.

***Trouble shooting***

If you forget to add the double quotes whilst trying to run the getdata function, you will get the error:

*Error in get(myprefix, envir = myglobal) : invalid first argument*

***Subset your data based on snp names***

To obtain a targetted subselection of snps, execute the function:

*subselectdata(snp\_names=snpnames*,*name2=TRUE)*

In which snpnames should be a vector with names of the snp files as listed in snps$name2 column (i.e.: chr\_pos). If you want to search with snps$name column, set name2 to FALSE.

***Subset your data based on population names***

To subselect individuals from particular populations, execute the function:

*subset\_pop(include\_pops=c("popname1","popname2","etc"))*

The include\_pops should be a vector with the names of the populations you want to select.

Run this function after the filterdata() function. Don't run the filterdata() function afterwards, because this will partially undo the changes.

To undo the changes, rerun the importdata() function.

***Subsample your data randomly***

Occasionally you might find that you want to randomly subsample your dataset, for example for to use as a control dataset. To sample your datasets, execute:

*subsampledata(nrinds=NULL,nrsnps=NULL,exportprefix=NULL)*

in which:

*nrinds:* how many individuals should subset contain? Should be less or equal to number of rows of inds dataframe.

*nrsnps:* how many loci should subset contain? Should be less or equal to number of rows of snps dataframe.

*exportprefix:* if NULL (default), data will be outputted as R objects.

If not NULL, data will be exported as PED and MAP files to SambaR\_output directory, and string will be used as prefix.

Output is a sambar list object called ‘mysambarsubset’.

***Find overlap between datasets***

Say you are working with two SNP datasets between two snp datasets generated by different protocols (e.g. different restriction enzymes) but using the same reference genome. You might wonder: do those datasets have SNPs in common? Because if so, you can combined both datasets and execute analyses on the combined dataset.

To find out, and to create the new combined dataset, SambaR comes with a function called findoverlap(). This functions expects two sambar list objects (created with the backupdata function), and if it finds overlapping snps, it outputs a new sambar list object called ‘mysambarcombined’. Say that your input sambar list objects are called mydata1 and mydata2, execute the following command:

*findoverlap(mydata1,mydata2,mycolours=c(“darkgreen”,”orange”,”blue”,”darkred”))*

You have to provide the colours of the populations, in alphabetical order of the populations names in the to be combined dataset. Just as a reminder: the default colours used in SambaR plots are, in order of occurence: blue, darkgreen, darkred, orange, purple (‘darkorchid4’), brown (‘#654321’), yellow, and darkgrey (‘gray20’).

To start working with the data, type:

*getdata(“mysambarcombined”)*

As mentioned above: don’t forget the double quotes!

Now you can run the SambaR functions again to do analyses on the combined dataset, starting with the filterdata() function.

**UNDER CONSTRUCTION**

**Add results from other software**

merger()

# 9. Manage your plots

***Change your plot colours***

The default colours used in SambaR plots are, in order of occurence: blue, darkgreen, darkred, orange, purple (‘darkorchid4’), brown (‘#654321’), yellow, and darkgrey (‘gray20’).

To use different (or more) colours in your plots, you have to assign new colours to your populations in the inds$popcol column. You can do so with the addcol()-function. This function expects as input a vector with colour names or codes. The length of this vector should equal the length of the populations vector.

If for example you have two populations and you want the first population to be shown in red and the second in blue, you should type:

*editcol(c("red","blue"))*

Execution of this function will update the inds$popcol column, and as such affect all plots created subsequently.

There are many online tools available which are helpful to select colour combinations. One example is colorbrewer2.org.

***Change your plot font type***

The default font type used by SambaR is R’s default font type, which is Helvetica (called ‘sans’), a sans serif font accepted by most journals. If you want or need to use another font, you might have to install this font first on your computer. This is for example true for Arial. To install all available fonts, run the following function:

*getfonts(importfonts=TRUE)*

This function uses the extrafont package to import and load font types. Once you have imported font types onto your computer (which can take up to 10 minutes on my computer), they will be stored safely until you shut down your computer. Every time you start a new R session, you do have to load the fonts though. This takes less than a second.

If you only want to load the fonts – without importing them – run the getfonts() function by setting the importfonts flag to FALSE (default value):

*getfonts(importfonts=FALSE)*

To subsequently set your desired font type, for example ‘serif’, type on the command line:

*mysambar$myfont <- “serif”*

All the functions you subsequently run, will generate plots with ‘serif’ as font type.

Note that many font types (especially the ones you imported) will only show up in the .pdf and .eps format. When trying to use these font types for the formats .png and .wmf you will get this non fatal warning:

*font family not found in Windows font database*

If so, the font type will default to Helvetica and the plots will still be generated.

Also note that SambaR currently does not embed the font types, meaning that if you send the plot to someone who has not installed the font type on his or her computer, it will not display properly. Journals will have the font types installed though, so this should not cause major problems.

**Importing pdf into Word file**

You can insert any pdf object into a word file by choosing ‘Adobe Acrobat Document’ under the option ‘Object’ in the ‘Insert’ tab (within Word). When you do so, make sure that the selected document is not opened in a PDF viewer, otherwise Word will choke. (If you happen to do so by mistake, (force) shutdown Word and try again after reopening the document.)

When inserting the file into word, the file might automatically be opened in a PDF viewer. If so, diagonal black lines will appear on the plot inserted in the Word document. The lines will disappear if you close the file in the PDF-viewer.

# 10. Understand your output

This section contains descriptions of the columns of the snps (10.1) and inds (10.2) dataframe, and describes some of the calculations underlying SambaR output plots (10.3).

## 10.1 The snps dataframe

Your snps dataframe could contain the following columns, depending on the functions you run and which options you used:

After running importdata() function:

*chr*: name of contig/chromosome/scaffold on which SNP was found (if no information in PED-file, 0)

*name*: SNP name

*morgan*: position in cM (if no information in PED-file, 0)

*pos*: position in bp (if no information in PED-file, 0)

*minor:* identity of minor allele (1, 2, 3, or 4)

*major:* identity of major allele (1, 2, 3, or 4)

*minor2:* identity of minor allele (A, C, T, or G)

*major2:* identity of major allele (A, C, T, or G)

*stackID :* locus number (if generated with STACKS)

*stackbp* : position of SNP on locus (if generated with STACKS)

*sameread:* is SNP located on same read as previous SNP? (if generated with STACKS)

*name2:* chr + name

*placed:* if reference genome is assembled to chromosomes, TRUE means that SNP is

found on a chromosome, and not an unplaced contig/scaffold. Defaults to TRUE if reference genome is not assembled to chromosome.

*autosomal:*  if reference genome is assembled to chromosomes, TRUE means that SNP is

found on a autosomal chromosome, and not an unplaced contig/scaffold and neither on a sex chromosome. Defaults to TRUE if reference genome is not assembled to chromosome.

*dist:* gap distance between SNP and previous SNP (in bp)

*dist2:* is SNP located on the same contig/scaffold/chromosome as previous SNP?

*samepos:* does SNP occur on same position as another SNP (if SNP dataset is generated with STACKS, this does occur occasionally)

*uniqpos:* inverse of samepos

*meandepth:* sequencing depth averaged over all samples (present if depthfile is present)

*depthfilter:* mean read depth of SNP below threshold? Threshold is defined as a bonferroni corrected right tail value, assuming mean loci specific read depths fit a normal distribution after removing outliers. Outliers are defined as the top 0.5% values. This filter has been introduced to filter out SNPs/reads with unusual high read depths.

After running the filterdata() function:

*GC:* major allele G or C?

*transit:* transition (A/G or C/T)?

*readpos:* position of SNP on read (same as *stackbp*)

*readpos2:* is SNP positioned outside dubious start/end of read? (see SNPsalongreads plot)

*misscount:* number of missing datapoints

*nonmissallelecount:* 2\*n\_individuals - misscount

*miss:* misscount/(misscount + nonmisscount)

*fmiss0.xx* miss < threshold?

*minorcount:* minor allele count

*majorcount:* major allele count

*maf:* minor allele frequency (minorcount/nonmisscount)

*maf\_pop:* minor allele frequency per population (with minor allele respective to all pops)

*maf2\_pop:* minor allele frequency per population (with minor allele respective to pop)

*hetero:*  heterozygosity

*AA/Aa/aa:* observed genotype counts

*expAA/expAa/expaa:* expected genotype counts (2\*maf\*(1-maf))

*heteropop:* heterozygosity per population

*hefilter:* maximum heterozygosity of He = 2\*maf\*(1-maf) + 0.5\*maf for maf > 0.05?

(An arbitrary threshold to filter out paralogs; see He\_vs\_maf.png)

*distfilter:* is gap to previous SNP bigger than certain threshold (default: 500 bp)?

*filter:* uniqpos + hefilter + (minorcount > 1) + fmiss0.xx + distfilter

*filter2:* uniqpos + hefilter + (minorcount > 1) + fmiss0.xx.

If you didn’t provide positional information about your snps (for example because you generated your data using STACKS denovo rather than STACKS refmap), filter2 will equal filter1.

*mafdiff\_pop:* difference in minor allele frequency between adjacent snps per population

After running the findstructure() function:

*Hexp\_meta:* 2\*maf\*(1-maf)

*F\_meta: (*Hexp\_meta – hetero)/Hexp\_meta

*HWEchi2*: Hardy Weinberg Equilibrium test chi-squared score

*HWE:* in Hardy Weinberg equilibrium?

*Hdeficit*: less heterozygotes than expected?

These columns are repeated with population specific estimates.

After running the calcdistance() function:

*WrightFst\_pop1\_pop2:* locus specific Wright Fst estimate

*WeirHe\_pop1\_pop2:* locus specific Weir & Cockerham heterozygosity estimate

*WeirFst\_pop1\_pop2:* locus specific Weir & Cockerham 1987 Fst estimate

After running the calcdiversity() function:

*mac\_pop:* minor allele count per population

## 10.2 The inds dataframe

Your inds dataframe could contain the following columns, depending on the functions you run and which options you used:

After running the importdata() function:

*name*: sample name

*pop/pop2*: population name (as defined in PED-file)

*popcol*: population colour (used in subsequent functions for plotting)

*meandepth*: sequencing depth averaged over all loci (present if depthfile is present)

*name2:* section until first period in name. If period is absent, first 10 characters.

After running the filterdata() function:

*miss:* proportion of missing data

*fmiss0.xx* miss < indmiss

*maf:* minor allele frequency averaged over segregating loci (respective to population to which sample belongs)

if maf = 0: all loci homozygous major allele

if maf = 0.5: either all loci are heterozygous, or 50% homozogous minor and

50% homozygous major, or anything in between

if maf = 1: all loci homozygous minor allele

*maf\_all:* minor allele frequency average over all loci (both segregating and non-segregating)

*nsegsites1:* number of segregating loci in population (same for all samples belonging to same population)

*nsegsites2:* number of segregating loci in population which are nonmissing in sample

*hetero:* number of heterozygous loci devided by number of segregating loci (respective to population to which sample belongs)

*hetero\_all:* number of heterozygous loci devided by total number of loci (both segregating and non-segregating)

*nsites1:* number of retained loci (same for all samples)

*nsites2:* number of retained loci which are nonmissing in sample

*expHe:* expected heterozygosity: 2\*maf\*(1 – maf)

*expHe\_all:* expected heterozygosity: 2\*maf\_all\*(1 – maf\_all)

*filter:* fmiss0.xx

*F:* inbreeding coefficient: (expHe – hetero)/expHe

*F\_all:* inbreeding coefficient: (expHe\_all – hetero\_all)/expHe\_all

After running the findstructure() function:

*pr\_pop*: Bayesian probability of a sample belonging to one of the predefined populations,

based on its genotype scores and population specific minor allele frequencies.

For example: say you have two loci and two populations (A and B). Population A has minor allele frequencies 0.5 and 0.25 for respectively locus 1 and locus 2, and population B has minor allele frequencies 0.1 and 0.05.

Individual1 has genotype scores 1 and 0, meaning this individual is heterozygous for the first locus and homozygous major allele for the second locus. The probability that this individual belongs to either population A or population B is calculated as follows:

Pr(A|ind1) = Pr(ind1|A)/(Pr(ind1|A)+Pr(ind1|B))

Pr(B|ind1) = Pr(ind1|B)/(Pr(ind1|A)+Pr(ind1|B))

We find:

Pr(ind1|A) = (2\*0.5\*0.5) \* (0.75\*0.75) = 0.5625

Pr(ind1|A) = (2\*0.9\*0.1) \* (0.95\*0.95) = 0.16245

Therefore:

Pr(A|ind1) = 0.5625/(0.5625 + 0.16245) = 0.776

Pr(B|ind1) = 0.5625/(0.5625 + 0.16245) = 0.224

No columns added by calcdistance() function.

After running the calcdiversity() function:

*hetero\_all2:* as hetero\_all, but calculated using different method

*theta:* expected proportion of differences between a haplotype from individual and haplotype from randomly chosen individual within same population

*autozygosity:* hetero\_all/theta

*theta2:* mean number of differences between two randomly drawn haplotypes within the population to which individual belongs

*harmonic\_number:* 1/1 + 1/2 + 1/3 + ... + 1/n-1 , with n defined as number of haplotypes (i.e. n\_individuals\*2 – 1)

*Watterson:* nsegsites1/harmonic\_number (Watterson’s estimate of theta2)

*TajimaD:* theta2 – Watterson

*theta2\_scaled:* theta2 per nucleotide

*Watterson\_scaled:* Watterson’s estimate per nucleotide

*rare\_alleles:* does the population to which the individual belongs, contain more or less

rare alleles than expected?

*TajimaD\_scaled:* Tajima’s D score per nucleotide

*genometheta:* genome wide estimate of theta: theta\*nsnps/nsites (nsites is explained elsewhere in this manual)

*genomehe:* genome wide estimate of heterozygosity: hetero\_all\*nsnps/nsites (nsites is explained elsewhere in this manual)

## 10.3 SambaR calculations

**SambaR estimates of number of (segregating) sites in inds dataframe**

The inds dataframe contains several columns which lists number of sites for each individual.

In the following description it is assumed that the data stored in mygenlight is saved in a matrix called mymatrix, and that only one row (one individual) is considered:

*mymatrix <- as.matrix(mygenlight)*

*myind <- mymatrix[i,]*

ALL SITES

***inds$nsites1***

Number of sites retained after filtering and thinning (same for each individual):

*length(myind[snps$filter])*

or

*nrow(snps[snps$filter,])*

***inds$nsites2***

Number of sites per individual retained after filtering and thinning, and after excluding missing data points:

*length(myind[snps$filter&!is.na(myind)])*

(Same as inds$nsites5 and inds$ndata)

***inds$nsites\_filter2***

Number of sites per individual retained after filtering (no thinning), and after excluding missing data points:

*length(myind[snps$filter2&!is.na(myind)])*

***inds$nsites\_nofilter***

Number of sites per individual retained after filtering on excessive heterozygosity only, and excluding missing data points:

*length(myind[snps$hefilter&!is.na(myind)])*

SEGREGATING SITES

For the next estimates, we also calculate for each SNP the number of minor allele copies in the population to which individual i belongs:

*mypopmaf <- glMean(mygenlight[inds$filter&inds$pop==popname,])*

***inds$nsegsites1***

Number of segregating sites per population after filtering and thinning (same for individuals belonging to the same population):

*length(myind[snps$filter&mypopmaf>0])*

***inds$nsegsites2***

Number of segregating sites per population after filtering and thinning, and after excluding missing datapoints:

*length(myind[snps$filter&mypopmaf>0&!is.na(myind)])*

***inds$nsegsites3***

Number of segregating sites per population retained after filtering on excessive heterozygosity only (same for individuals belonging to the same population):

*length(myind[snps$hefilter&mypopmaf>0])*

(These estimates are displayed in the left tile of the plot pi.pop.stripchart and pi.pop.vioplot.)

***inds$nsegsites\_filter2***

Number of segregating sites per population after filtering (no thinning) , and after excluding missing datapoints:

*length(myind[snps$hefilter2&mypopmaf>0&!is.na(myind)])*

***inds$nsegsites\_nofilter***

Number of segregating sites per population retained after filtering on excessive heterozygosity only , and after excluding missing datapoints:

*length(myind[snps$hefilter&mypopmaf>0&!is.na(myind)])*

(Same as inds$nsegsites4)

**SambaR’s calculation of pairwise sequence dissimilarity, nucleotide diversity (π), Dxy, Watterson’s theta (θW) and Tajima’s D illustrated using an example dataset**

Implemented in SambaR is the function ‘calcpi’, which is invoked by several of SambaR’s seven main functions. This function calculates observed pairwise sequence dissimilarities, followed by several population-genetic measures which depend on these estimates. Here we will describe the algorithm of the calcpi function using a small example dataset.

Consider the following genotype dataset for 3 individuals and 2 SNPs (in which 0 codes for homozygous major, 1 for heterozygous, 2 for homozygous minor, and NA for missing data):

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | ***SNP1*** | ***SNP2*** |
| ***PopA*** | ***Ind1*** | 0 | 1 |
| ***PopA*** | ***Ind2*** | 2 | NA |
| ***PopB*** | ***Ind3*** | 1 | 0 |

***Pairwise sequence dissimilarity***

SambaR calculates the number/proportion of differences and missing data points between and within individuals – taking into account that each sample pair provides four potential sequence comparisons (AB,Ab,aB,ab) – and subsequently estimates sequence dissimilarity, both for ‘between individual’-comparisons and ‘within individual’-comparisons:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***n\_***  ***comparisons*** | ***n\_differences*** | | | | ***n\_missing\_data\_points*** | | | | ***pairwise sequence dissimilarity*** |
| ***SNP1*** | ***SNP2*** | ***total*** | ***mean (O)*** | ***SNP1*** | ***SNP2*** | ***total*** | ***mean***  ***(M)*** | ***O/(nsnps-M)*** |
| ***Ind1-Ind2*** | 4 | 4 | NA | 4 | 1 | 0 | 4 | 4 | 1 | 1/(2-1)=1 |
| ***Ind1-Ind3*** | 4 | 2 | 2 | 4 | 1 | 0 | 0 | 0 | 0 | 1/(2-0)=0.5 |
| ***Ind2-Ind3*** | 4 | 2 | NA | 2 | 0.5 | 0 | 4 | 4 | 1 | 0.5/(2-1)=0.5 |
| ***Ind1*** | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1/(2-0)=0.5 |
| ***Ind2*** | 1 | 0 | NA | 0 | 0 | 0 | 1 | 1 | 1 | 0/(2-1)=0 |
| ***Ind3*** | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1/(2-0)=0.5 |
| ***total*** | 15 |  |  | 12 |  |  |  | 9 |  |  |

***Nucleotide diversity (π)***

Nucleotide diversity (π), which is the expected proportion of differences between any two randomly drawn sequences, is calculated as:

π = sum(n\_differences)/(n\_comparisons \* n\_snps – sum(n\_missing\_data\_points)).

For the overall example dataset, π equals: *12/(15\*2 – 9) = 12/21 = 0.5714.*

SambaR also estimates ***π*** per population:

*πpopA = (4 + 1 + 0)/(6\*2 – 5) = 5/7 = 0.714*

*πpopB = 0.5*

***Dxy***

SambaR obtains an estimate of *Dxy*, the mean proportion of differences between two sequences randomly drawn for two different populations, by calculating the mean number of pairwise sequence dissimilarity over all possible sample pairs.

For the example dataset, Dxy for pop1 and pop2 is obtained from averaging the pairwise sequence dissimilarity estimates obtained for Ind1-Ind3 and Ind2-Ind3:

*n\_differences = 4 + 2 = 6*

*n\_datapoints = 8\*2 = 16*

*n\_missing\_data\_points = 0 + 4 = 4*

*Dxy = n\_differences/(ndatapoints - n\_missing\_data\_points) = 6/(16 – 4) = 0.5*

***Fstπ***

Fstπ = (πbetween\_pop – πwithin\_pop )/ πbetween\_pop =(Dxy – (∑πpop)/npops) /Dxy

Fstπ for population pair popA and popB in the example dataset equals (see also the sections on nucleotide diversity and Dxy):

*Fstπ = (Dxy – (∑πpop)/npops) /Dxy = (0.5- (0.714+0.5)/2)/0.5 = -0.214*

Negative Fstπ values indicate that individuals within populations are, in terms of their genotypes, more different from each other than individuals from different populations.

***Watterson’s theta*** ***(θW)***

SambaR calculates Watterson's estimate of theta (θW) by dividing the number of segregating sites within a population by the harmonic number (an). The number of segregating sites (S) is defined as the number of SNPs with a minor allele frequency above 0. The average number of sequences (n), needed to calculate the harmonic number (an), is estimated as twice the number average number of individuals with non-missing data per SNP.

For the example dataset, Watterson’s theta for all individuals combined equals:

*S = 2*

*n = 2\*(3 + 2)/2 = 5*

*an = ∑i=1n-1 1/(1/i) = 1/1+1/2+1/3+1/4 = 2.083333*

*θW = 2/2.083333 = 0.96*

Hence for each pair of sequences (consisting of 2 SNPs) 0.96 sites are expected to differ (assuming neutrality). SambaR reports estimates per site (θW\_persite), which equals 0.48 for the numerical example.

***Tajima’s D***

SambaR calculates Tajima’s D scaled per single site as:

D = π - θW\_persite

For the example dataset, SambaR’s Tajima’s D estimate (for all individuals combined) equals: *0.5714 – 0.48 = 0.0914*

SambaR also calculates significance using a chi squared test on observed and expected number of differences. For the example dataset (all individuals combined) the calculation becomes:

*n\_expected = (ndatapoints - n\_missing\_data\_points)\*θW\_persites = (15\*2 – 9)\*0.48 = 10.08*

*X2 = (O – E)2/E = (n\_differences – n\_expected)2/n\_expected = (12 – 10.08) 2/10.08 = 0.37*

Next SambaR uses the R base function pchisq(df=1,lower.tail=FALSE) to derive the corresponding p-value:

*P(X2 = 0.37,df=1) = 0.55*

**SambaR’s estimation of genome wide heterozygosity**

If users provide to the nrsites flag of SambaR’s calcdiversity function (default is NULL), an estimate of the total number of sequenced sites from which the SNP data is derived, SambaR will estimate genome wide estimates (e.g. genome wide heterozygosity and genome wide nucleotide diversity).

For RADseq data, the number of sequenced sites, here denoted as ‘N\_total’, equals the combined length of all polymorphic as well as monomorphic stacks which passed filter settings, and can be obtained from the sumstats summary file.

For resequencing data, N\_total equals the total number of monomorphic and polymorphic sites of the filtered data set. This number can for example be obtained by counting the number of lines in the VCF file, excluding the header section. The precondition is that the VCF file contains both monomorphic and polymorphic sites. If the VCF file is generated with bcftools, users should not include the -v option when running the bcftools calls command, otherwise monomorphic sites will not be outputted.

SambaR calculates genome wide heterozygosity (Hegenome) and proportion of segregating sites (Sgenome) as:

*Hegenome = (nH/nind \* nsnps)/ntotal*

*Sgenome = nseg/ntotal*

In which nH denotes number of heterozygous sites observed for individual i within the SNP dataset, nind denotes the number of non-missing data points for individual i within the SNP dataset, nsnps denotes the total number of snps in the dataset, nseg denotes number of segregating sites for population j, and ntotal denotes the total number of sequenced sites (sum of total length of both monomorphic and polymorphic STACKS loci. All calculations are based on the full SNP dataset, excluding SNPs with excessive heterozygosity.

The method is illustrated in the scheme below:

**Genome**

**110 bp STACKs loci**

green: polymorphic

blue: monomorphic

red: SNP

combined length (ntotal):

5x110bp = 550 bp

Pop1 Ind1 A A C C **SNP data**

A G C T

Pop1 Ind2 A A C N black: fixed

A G G N red: segregating

Pop2 Ind3 A A C T orange: out of HWE,

T G C T excluded from calculations

Pop2 Ind4 A A C C

T G G T

Pop2 Ind5 A A N C

A G N T

Because one SNP is excluded from the analysis because of HWE considerations, nsnps = 3.

Heterozygosity and segregating site calculations are displayed in black and grey respectively:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | all sites  non-missing  nind | segregating sites  within population  nseg | segregating  sites non-missing  nind2 | Heter-ozygous sites  nH | Heterozygosity within SNP dataset\*  HSNPS =  nH /nind | Heterozy-gosity for segregating sites\*\*  Hseg =  nH /nind2 | Genome wide  Proportion  Segregating sites  S = nseg/ntotal | Genome wide heterozygosity  (Hsnps\*nsnps)/  ntotal |
| Pop1 | Ind1 | 3 | 2 | 2 | 1 | 1/3 = 0.333 | 1/2 = 0.5 | 0.0036 | 0.0018 |
| Pop1 | Ind2 | 2 | 2 | 1 | 1 | 1/2 = 0.5 | 1/1 = 1 | 0.0036 | 0.0027 |
| Pop2 | Ind3 | 3 | 3 | 3 | 1 | 1/3 = 0.333 | 1/3 = 0.333 | 0.0055 | 0.0018 |
| Pop2 | Ind4 | 3 | 3 | 3 | 3 | 3/3 = 1 | 3/3 = 1 | 0.0055 | 0.0055 |
| Pop2 | Ind5 | 2 | 3 | 2 | 1 | 1/2 = 0.5 | 1/2 = 0.5 | 0.0055 | 0.0027 |

*\*This estimate is stored in the column inds$hetero\_all, and is identical to the multi-locus heterozygosity (MLH) estimates returned by the function MLH of the R package ‘inbreedR’, stored in the column inds$MLH.*

*\*\*This estimate is stored in the column inds$hetero. These estimates are higher than (or equal to) the values in the inds$hetero\_all column, because non-segregating sites are excluded from the calculation.*

**SambaR’s estimation of locus specific Fst-values, illustrated using an example dataset**

Differences in minor allele frequencies (MAF, or p) between populations can be summarized using a Fst-metric. SambaR relies upon the stamppFst function (of the package ‘Stampp’) to generate Weir & Cockerham 1984 Fst estimates, along with associated significance values. This function generates for each population pairwise comparison one estimate for all SNPs combined.

In addition, SambaR uses built-in functions to calculate Fst estimates for each single SNP. SambaR generates three different Fst measures:

* Fst = Var(p)/(p(1-p)) Wright, 1943
* Fst = (HT-HS)/HT Nei, 1977
* Fst = (f0 – f1)/(1 – f1) Cockerham & Weir, 1987

For illustrative purposes we will consider again the following genotype dataset for 3 individuals and 2 SNPs (in which 0 codes for homozygous major, 1 for heterozygous, 2 for homozygous minor, and NA for missing data):

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | ***SNP1*** | ***SNP2*** |
| ***PopA*** | ***Ind1*** | 0 | 1 |
| ***PopA*** | ***Ind2*** | 2 | NA |
| ***PopB*** | ***Ind3*** | 1 | 0 |

***Wright 1943 Fst***

This measure was first published by Sewall Wright in 1943 (Wright, 1943, see equation 48 and the following lines).

*Var(p)*, also denoted as sp2, or σp, denotes the variance in minor allele frequencies among populations. If p denotes the minor allele frequency, and if subscripts 1 and 2 denote either of the two populations considered in a pairwise population comparison, this estimate can be described as:

*Var(p)* = ½\*((p1 – p)2 + (p2 – p)2)

To standardize this estimate (i.e. make it comparable among SNPs regardless of their mean minor allele frequencies), the value it divided by the maximum value. It can be mathematically shown that for population pairs which are fixated for different alleles (i.e. p1=1 and p2=0 or vice versa), *Var(p)* equals: p(1-p). Therefore a standardized estimate is given by:

*Fst = Var(p)/**(p(1-p))*

For SNP2 in the example dataset above, the Var(p) and Fst estimates equal:

*Var(p)* = ½\*((0.5 – 0.25)2 + (0 – 0.25)2) = 0.0625

*(p(1-p)) =* 0.25\*0.75 = 0.1875

*Fst = Var(p)/(p(1-p))* = 0.0625/0.1875 = 1/3

***Nei 1977 Fst***

This measure was developed by Nei (1977) for multi-allelic loci.

*HT* denotes the expected heterozygosity in the overall dataset (total population).

*HS* denotes the expected heterozygosity in the subpopulations.

If p denotes the minor allele frequency, and if subscripts 1 and 2 denote either of the two populations considered in a pairwise population comparison, these estimates are equal to:

*HT* = 2\*p\*(1-p)

*HS* = ½\*(2\*p1\*(1-p1) + 2\* p2\*(1-p2))

For SNP2 in the example dataset above, the HT and HS estimates are equal to:

*HT* = 2\*0.25\*(1-0.75) = 0.125

*HS* = ½\*(2\*0.5\*(1-0.5) + 2\*0\*1) = 0.25

Fst = (HT-HS)/HT = (0.25 – 0.125)/(0.25) = 1/2

***Cockerham & Weir 1987 Fst***

This measure was published by Cockerham and Weir in 1987, and is not be confused with their Fst metric published in 1984. The original notation used by Cockerham and Weir (1987) was: Fst = (θ1 – θ2)/(1 – θ2). Here we denote θ1 as f0 and θ2 as f1.

*f0* denotes the probability that two randomly drawn alleles from within a (sub)population are identical.

*f1* denotes the probability that two randomly drawn alleles from the overall population (the total or metapopulation) are identical.

If p denotes the minor allele frequency, and if subscripts 1 and 2 denote either of the two populations considered in a pairwise population comparison, these estimates are equal to:

*f0* = ½\*(p12\*(1-p1)2 + p22\*(1-p2)2)

*f1* = p1\*p2 + (1-p1)\*(1-p2)

For SNP2 in the example dataset above, the f0 and f1 estimates are equal to:

*f0* = ½\*(0.52 + (1-0.5)2 + 02 + (1-0)2) = 0.75

*f1* = 0.5\*1+ 0.5\*0 = 0.5

Hence, there is a 75% probability of drawing two identical alleles when pooling from within one of the populations, compared to a 62.5% probability when drawing two alleles from the metapopulation.

The Fst estimates for SNP2 becomes:

Fst = (f0 – f1)/(1 – f1) *=* (0.75 – 0.5)/(1-0.5) = 0.5

Note that from its definition it follows that *f1* is an estimate of expected homozygosity in the populations in the absence of population structure. Expected heterozygosity is therefore simply:

*He* = 1 – f1

The ‘selectionanalyses’-function of SambaR will generate Fdist plots, showing locus specific Fst estimates on the y-axis against He (= 1 – f1) estimates on the x-axis. If SNPs are biallelic, and if datasets consists of two populations/groups, this spectrum of possible He-FST values has the shape of a shark fin, of which the left boundary is described by FST = He and represents loci which are segregating in one population only. The right boundary of the ‘shark fin’-spectrum represents loci with opposing allele frequencies in either population (e.g 0.3-0.7 in one population and 0.7-0.3 in the other population), and is described by FST = (2He-1)/He for He>=0.5.

**SambaR’s estimation of inbreeding coefficient F**

SambaR uses two methods to calculate individual inbreeding coefficient F, the probability that the two alleles at any locus of a diploid individual are identical by descent (IBD): FH (Kardos et al., 2015) and the here defined Fπ (see below).

***FH***

F can be defined as the proportion of expected heterozygous sites (in case of no inbreeding) which are homozygous due to inbreeding. In formula:

*FH = (obshomo - exphomo)/(n – exphomo)* (Kardos et al., 2015)

In which:

n: total number of sites (for an individual), excluding missing data points

obshomo: observed number of homozygous sites (for an individual)

exphomo: expected number of homozygous sites (for an individual), estimates as:

*∑in qi2+(1-qi)2*

in which qi denotes the minor allele frequency in the population to which the individual belongs, which is derived from all individuals in the population except for the individual for which F is calculated.

SambaR evaluates significance by applying a chi-squared test on observed and expected number of homozygous sites.

Consider, as an example, a diploid individual A which is genotyped at 10 sites, of which 6 are heterozygous. Through selfing individual A produces individual B. The most likely genotype of individual B consists of (4 + 0.5\*6 =) 7 homozygous sites.

If all other individuals in the population are not inbred, and on average have the same number of homozygous and heterozygous sites as individual A, then the expected number of homozygous sites equals 4. The inbreeding coefficient of individual B therefore equals:

*FH = (obshomo - exphomo)/(n – exphomo) = (7 - 4)/(10 - 4) = 3/6 = 0.5*

The *FH* value of 0.5 correctly indicates that individual B results from selfing. It also correctly indicates the proportion of autozygous sites of individual B (i.e. proportion IBD). Assuming the homozygous sites of individual A are IBS (identical by state) but not identical by descent (IBS), the most likely number of autozygous sites of individual B equals (0.5\*10 =) 5.

Significance is evaluated using a chi-squared test:

*X2 = (O – E)2/E = (obshomo – exphomo)2/exphomo = (7 – 4) 2/4 = 2.25*

SambaR uses the R base function pchisq(df=1,lower.tail=FALSE) to derive the corresponding p-value:

*P(X2 = 2.25,df=1) = 0.1336144*

***F***π

The second method SambaR uses to estimate heterozygosity is based on a comparison between the heterozygosity (He) of an individual (j) and the mean pairwise sequence dissimilarity (here defined as πj) of this individual with all other individuals in the population:

*Fπ = 1-Hej/πj*

SambaR evaluates significance by applying a chi-squared test on observed and expected number of heterozygous sites, in which the number of expected heterozygous sites is defined estimated by πj times the number of sites.

Consider, as an example, an individual with a heterozygosity of 0.23 and a mean pairwise dissimilarity with other individuals of the population of 0.25. SambaR estimates the inbreeding coefficient for this individual as:

*Fπ = 1-0.23/0.25 =0.08*

If the heterozygosity and nucleotide diversity estimates are based on a dataset of 1000 sites, a chi-squared test returns:

*X2 = (O – E)2/E = (Hej – πj)2/πj = (230 – 250) 2/250 = 1.6*

SambaR uses the R base function pchisq(df=1,lower.tail=FALSE) to derive the corresponding p-value:

*P(X2 = 1.6,df=1) = 0.2059032*

**SambaR’s Bayesian population assignment (BPA) test, illustrated using an example dataset**

The BPA test addresses the question: among a set of predefined populations, which population is most likely to be the origin of a particular individual? The BPA test calculates the probability that an individual originates from each of the predefined populations, given the observed genotype of the individual and given the observed population allele frequencies.

Consider, as an analogy, two jars. Jar A contains 10 red and 90 white balls. Jar B contains 1 red and 99 white balls. A ball is drawn from one of these jar, and the ball is red. From which bottle has the ball been drawn? According to Bayes Rule, the posterior probabilities are:

Pr(A|red) = Pr(red|A)/(Pr(red|A)+Pr(red|B)) = 0.1/(0.1+0.01) = 0.91

Pr(B|red) = Pr(red|B)/(Pr(red|A)+Pr(red|B)) = 0.01/(0.1+0.01) = 0.09

Hence, there is a 91% probability the ball has been drawn from jar A, and a 9% probability the ball has been drawn from jar B.

The BPA test uses a similar algorithm to calculate the probabilities that a sample has been drawn from each of the set of predefined populations:

Let Hkj denote the probability that individual k belongs to population j, let MAFij denote the minor allele frequency of locus i in population j (excluding the genotype of individual k), let Oi denote the genotype of individual k for locus i (i.e. number of minor alleles (0, 1, or 2)), and let k denote the number of predefined populations. For any given locus, the probability that an individual belongs to population j can be described as:

Pr(Hkj|Oi) = (Pr(Oi|Hkj)\*Pr(Hkj))/Pr(Oi) {Oi = 0,1,2 and 0 ≤ MAFij ≤ 1}

in which:

Pr(Oi=0|Hkj) = (1-MAFij)2

Pr(Oi=1|Hkj) = 2\*MAFij\*(1-MAFij)

Pr(Oi=2|Hkj) = MAFij2

If assuming a flat prior distribution (i.e. Pr(Hkj) = 1/j), the formula for Pr(Hkj|Oi) can be simplified to:

Pr(Hkj|Oi) = Pr(Oi|Hkj) / ∑jk Pr(Oi|Hkj)

To calculate the probability for the nth locus, SambaR uses a recursive formula:

Pr(Hkj|On) = (n-1)\*Pr(Hkj|On-1)\*Pr(Hkj|Oi)

As an example, consider the following genotype dataset for 10 individuals and 2 SNPs (in which 0 codes for homozygous major, 1 for heterozygous, 2 for homozygous minor, and NA for missing data):

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | ***SNP1*** | ***SNP2*** |
| ***PopA*** | ***Ind1*** | 2 | 2 |
| ***PopA*** | ***Ind2*** | 0 | 1 |
| ***PopA*** | ***Ind3*** | 1 | 0 |
| ***PopA*** | ***Ind4*** | 0 | 0 |
| ***PopA*** | ***Ind5*** | 0 | 2 |
| ***PopB*** | ***Ind6*** | 1 | 2 |
| ***PopB*** | ***Ind7*** | 2 | 2 |
| ***PopB*** | ***Ind8*** | 2 | 2 |
| ***PopB*** | ***Ind9*** | 1 | 1 |
| ***PopB*** | ***Ind10*** | 2 | 0 |

All individuals have been assigned to a set of two populations, PopA or PopB. Ind1 has been assigned to PopA, but what is the probability it does indeed belong to PopA (assuming the other population assignments are correct)?

The population minor allele frequencies, not considering Ind1, are:

|  |  |  |
| --- | --- | --- |
|  | ***SNP1*** | ***SNP2*** |
| ***PopA*** | 1/8 = 0.125 | 3/8 = 0.375 |
| ***PopB*** | 8/10 = 0.8 | 7/10 = 0.7 |

For both SNPs, Ind1 is homozygous for the minor allele (i.e. genotype 0).

Let Pr\_sum denote: (Pr(2|popA)+Pr(2|popB).

The posterior probabilities for SNP1 and SNP2 are equal to:

SNP1:

*Pr(popA|2) = Pr(2|popA)/Pr\_sum = 0.1252/(0.1252+0.82) = 0.02383222*

*Pr(popB|2) = Pr(2|popB)/Pr\_sum = 0.82/(0.1252+0.82) = 0.9761678*

SNP2:

*Pr(popA|2) = Pr(2|popA)/Pr\_sum = 0.3752/(0.3752+0.72) = 0.2229931*

*Pr(popB|2) = Pr(2|popB)/Pr\_sum = 0.72/(0.3752+0.72) = 0.7770069*

The combined posterior probabilities are equal to:

*Pr(popA|2 2) = (0.02383222\*0.2229931)/(0.02383222\*0.2229931+0.9761678 \* 0.7770069) = 0.006957837*

*Pr(popB|2 2) = (0.9761678 \* 0.7770069 /(0.02383222\*0.2229931+0.9761678 \* 0.7770069) = 0.9930422*

In conclusion, given the observed genotype of Ind1 and the observed allele frequencies in PopA and PopB, Ind1 belongs with 99.3% probability to PopB and with 0.7% probability to PopA, suggesting that the original population assignment is incorrect.

**The ‘distinct clustering’-score (dc-score) algorithm, illustrated using an example dataset**

SambaR aims to facilitate objective interpretation of ordination analyses by calculating a ‘dc-score’ for PCA, PCoA, CA and DAPC analyses. The dc-score, or ‘distinct clustering’-score, measures the overlap between population clusters in a 2-dimensional space defined by the first and second axis of an ordination analysis (e.g. PCA and PCoA).

The algorithm behind the dc-score is as follows:

Let p1i\_j and p2i\_j denote the loading of sample i, belonging to population j, on the first and second ordination axes, and let nj denote the number of individuals belonging to population j.

The mean loadings (population centres) for population j are defined as:

p1j = ( ∑inj p1i\_j )/nj

p2j = ( ∑inj p2i\_j )/nj

The mean distance dj of samples belonging to population j from the population centre of population j is defined as:

d1i\_j = | p1i\_j - p1j |

d2i\_j = | p2i\_j - p2j |

di\_j = √((d1i\_j)2 + (d1i\_j)2)

dj = ( ∑inj  di\_j )/nj

The mean distance of all samples from their population centres, given nz populations, is defined as:

dz = ( ∑jz  dj )/nz

The distance between population centres for population pair k, dk, is defined as:

d1k = | p1jj=1 - p1jj=2 |

d2k = | p2jj=1 - p2jj=2 |

dk = √((d1k)2 + (d2k)2)

The mean distance between population centres for all nm populations pairs, dm, is defined as:

dm = ( ∑km  dk )/nm

The dc-score is defined as:

dc = dz/dm

As an example, consider an ordination analysis on six samples (from three populations) returning the following output:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | ***axis1*** | ***axis2*** |
| ***PopA*** | ***Ind1*** | -0.02 | -0.03 |
| ***PopA*** | ***Ind2*** | 0.02 | -0.03 |
| ***PopB*** | ***Ind3*** | 0.03 | 0.03 |
| ***PopB*** | ***Ind4*** | 0.02 | 0.02 |
| ***PopC*** | ***Ind5*** | -0.03 | 0.03 |
| ***PopC*** | ***Ind6*** | -0.01 | 0.01 |

To calculate the dc-score, SambaR derives the mean loadings per population (‘population centres’):

|  |  |  |
| --- | --- | --- |
|  | ***axis1*** | ***axis2*** |
| ***PopA*** | 0 | -0.03 |
| ***PopB*** | 0.025 | 0.025 |
| ***PopC*** | -0.02 | 0.02 |

Thirdly, SambaR calculates the mean distance between sample loadings and population centres, and between population centres, and calculates the dc-score as the ratio between both distances:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | ***axis1*** | ***axis2*** | ***distance*** |  |  | ***distance*** | ***dc (dz/dm)*** |
| ***PopA*** | ***Ind1*** | -0.02 | -0.03 | 0.02 |  | ***popA-popB*** | 0.06041523 |  |
| ***PopA*** | ***Ind2*** | 0.02 | -0.03 | 0.02 |  | ***popA-popC*** | 0.05385165 |  |
| ***PopB*** | ***Ind3*** | 0.03 | 0.03 | 0.007071068 |  | ***popB-popC*** | 0.04527693 |  |
| ***PopB*** | ***Ind4*** | 0.02 | 0.02 | 0.007071068 |  |  |  |  |
| ***PopC*** | ***Ind5*** | -0.03 | 0.03 | 0.014142136 |  |  |  |  |
| ***PopC*** | ***Ind6*** | -0.01 | 0.01 | 0.014142136 |  |  |  |  |
| ***mean*** |  |  |  | dz = 0.01373773 |  |  | dm = 0.05318127 | 0.258319 |

**SambaR’s calculation of the folded site frequency spectrum (SFS), illustrated using an example dataset**

SambaR uses two steps to generates folded site frequency spectrum (SFS) vectors:

* The first step involves counting for each SNP and for each population the number of minor allele copies. This counting is performed using the glSum function of the adegenet package. The minor allele is defined based on the total dataset. As a consequence, some populations can carry SNPs with more minor allele copies than major allele copies. If so, SambaR subtracts for these particular SNP the number of minor allele copies from the total number of observed allele copies (ignoring missing data points). This latter number is computed with the glNA-function of the adegenet package.
* The second step consists of binning and counting SNPs based on their number of minor allele copies.

By default, Sambar includes all SNPs in the calculation, also SNPs which did not pass filter settings. In contrast, it does not include individuals which did not pass the filter settings. The SFS vector generated by SambaR does not include sites with zero copies of the minor allele. As a result, the length of the SFS-vector for each population equals the number of retained individuals within this population.

Consider, as an example, the following genotype dataset for 7 individuals and 4 SNPs (in which 0 codes for homozygous major, 1 for heterozygous, 2 for homozygous minor, and NA for missing data):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | ***SNP1*** | ***SNP2*** | ***SNP3*** | ***SNP4*** |
| ***PopA*** | ***Ind1*** | 1 | 1 | 0 | NA |
| ***PopA*** | ***Ind2*** | 2 | 0 | 0 | 0 |
| ***PopA*** | ***Ind3*** | 2 | 0 | 0 | 2 |
| ***PopA*** | ***Ind4*** | 0 | 0 | 0 | 0 |
| ***PopB*** | ***Ind5*** | 1 | 1 | 2 | 0 |
| ***PopB*** | ***Ind6*** | 0 | 2 | 2 | 1 |
| ***PopB*** | ***Ind7*** | 0 | 0 | 2 | 0 |

***Step 1. Count number of minor allele copies***

Counting the number of minor allele copies results in the following data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***SNP1*** | ***SNP2*** | ***SNP3*** | ***SNP4*** |
| ***PopA*** | 8-5=3\* | 1 | 0 | 2 |
| ***PopB*** | 1 | 3 | 6-6=0\* | 1 |

\*Due to population structure, it can occur that within a population the minor allele (which is defined relative to the metapopulation) has in fact more copies than the minor allele. If that is the case, for this population and for this particular SNP the number of major alleles copies are counted (rather than number of minor allele copies).

**Step 2. Binning SNPs based on their number of minor allele copies**

The second step is to bin and count the SNPs based on their number of minor allele copies present in either population. The following data frame shows the number of SNPs for each bin class:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***Number of minor allele copies*** | | | |
|  | ***1*** | ***2*** | ***3*** | ***4*** |
| ***PopA*** | 1 | 1 | 1 | 0 |
| ***PopB*** | 2 | 0 | 1 |

Hence the site frequency spectra are:

PopA: 1,1,1,0

PopB: 2,0,1

Note that the row sums are equal to the number of segregating sites within either population. The length of the vectors is equal to the number of individuals within either population.

# 11. Simulate your data

**outliersim()**

SambaR provides a function, called outliersim, to simulate allele frequencies of neutral and adaptive SNPs in founder populations which are separated in the past from their source population due to a vicariance event, with no possibility for gene flow. The main purpose of these simulations is to estimate the power and specificity of the selection scans GWDS, PCadapt and OutFLANK given a user defined demographic model. It is assumed that all SNPs are unlinked, and that allele frequencies are affected by drift only (neutral alleles) or by a combination of drift and selection (adaptive alleles).

The general demographic model consists of a (diploid) source population which at t = 0 buds of one or two founder populations:

t=0 time (generations) t=*ngen* *samplesize*

*nefounderpop1* **Founder population 1 sample**

*nfounders*

*nesourcepop* **Source population** **sample**

*nfounders*

*nefounderpop2* **Founder population 2 sample**

By default the simulations tool will run the simulations for both founder populations, and report three different outcomes: source vs FP1, source vs FP2, and source vs FP1 and FP2 combined. If the user is interested one founder population only, the results for FP2 and the combined output can simply be ignored.

The user can define constant effective population sizes of the source and founders population (*nesourcepop, nefounderpop1, nefounderpop2),* the number of founders (*nfounders*), the number of generations between the founder/vicariance event and the time of sampling (*ngen*), the sample size (number of individuals) per population (*samplesize*), and the number of SNPs (*nloc*).

The demographic model does not allow for gene flow nor time-variant population sizes.

Optionally, the user can define a number of SNPs (*nselectedloci*) which experience positive selection in both founder populations with a user defined selection strength (*selcoef*). If the flag do\_selectionscan is set to TRUE, selection scans are executed on the simulated data, and the power and specificity of the selection scans are estimated.

An example command:

*outliersim(export=TRUE,nloc=10000,ngen=1500,nesourcepop=10000,nefounderpop1=5000,nefounderpop2=5000,nfounders=500,selcoef=0.01,samplesize=30,nselectedloci=100,do\_selectionscan=TRUE,do\_plotpvalues=FALSE,mycorrection=’none’)*

The flag mycorrection can be used to specify the desired multiple test correction method, and accepts the values ‘none’, ‘bonferroni’, ‘holm’ and ‘BH’ (Benjamini-Hochberg).

The manhattan plot generated by outliersim, shows GWDS scores. All SNPs above the threshold (dashed line) are marked as outlier by GWDS. All points in green are marked by PCadapt as outliers. All points in orange are marked by OutFLANK as outliers.

**runsim\_power()**

The outliersim function allows to estimate power and specificity of the selection scans GWDS, PCadapt and OutFlank for one demographic model. SambaR also provides a function, runsim\_power, to generate these estimates for a range of scenarios with varying founder Ne and selection coefficient strengths. To execute this function, run:

*runsim\_power(n\_gen=50,n\_loci=1000,mycorr="bonferroni",n\_selectedloci=100,selcoefvector=c(0.05,0.1),nevector=c(50,100),do\_export=TRUE,dohaploid=TRUE,my\_comparison=1)*

The runsim\_power runs the outliersim function for all combinations of founder Ne and selection strength (defined with the nevector and selcoefvector flags) and afterwards displays all power and specificity estimates in heatmaps. By default the power and specificity scores are shown for the pairwise comparison between the source population and founder population 1. Set the flag ‘my\_comparison’ to 2 to obtain the scores for the pairwise comparison between the source population and founder population 2. Set the flag ‘my\_comparison’ to 3 to obtain the scores for the comparison between the source population and both founder populations combined.

The flag mycorrection can be used to specify the desired multiple test correction (MTC) method, and accepts the values ‘none’, ‘ ‘bonferroni’, ‘holm’ and ‘BH’ (Benjamini-Hochberg).

**plotfdr()**

Another option is to create line plots showing GWDS, OutFLANK and PCadapt power and specificity estimates on the y-axis and founder effective population sizes on the x-axis, for the three different MTC methods combined. These plots make it easier to compare the test performances, although the downside is that the plots can show the results for one selection coefficient only (default 0.1). To run the analysis, execute the command:

*plotfdr(do\_analysis=TRUE,do\_export=TRUE,loci\_nr=10000,gen\_nr=20,selected\_nr=1000,vector\_ne=c(20,40,60,80,100,120,160,200),samples\_nr=30,selection\_coefficient=0.1,plot\_fdr=FALSE,add\_bh=TRUE,myinputmatrix=NULL,my\_comp=1)*

For precise estimates, set loci\_nr to 100.000 and selected\_nr to 10.000.

By default the power and specificity scores are shown for the pairwise comparison between the source population and founder population 1. Set the flag ‘my\_comp’ to 2 to obtain the scores for the pairwise comparison between the source population and founder population 2. Set the flag ‘my\_comp’ to 3 to obtain the scores for the comparison between the source population and both founder populations combined.

The plotfdr function will export into the working directory both an output plot and output tables with the power and specificity estimates. If you set the flag plot\_fdr to TRUE, the output plot will also show false discovery rate estimates for the simulated data. However, these FDR estimates are not really useful, because the true false discovery rate for your empirical dataset depends on the proportion of SNPs which are under selection. This is an unknown parameter, but likely not 10%, as implied by the example command above. However, if you rerun the function, but this time with the flag do\_analysis set to FALSE and with providing to the myinputmatrix flag the name of the output files (without the MTC name suffix), a new plot will be generated which shows the false discovery rate given various proportions of adaptive SNPs:

*plotfdr(do\_analysis=FALSE,do\_export=TRUE,loci\_nr=10000,selected\_nr=1000,vector\_ne=c(20,40,60,80,100,120,160,200),samples\_nr=30,plot\_fdr=FALSE,myinputmatrix="FDR\_vs\_correctionmethod.comp3.s0.1.nSNPs100000.samplesize30.ngen20")*

The FDR estimates will be based on the Bonferroni correction.

To display power and specificity estimates excluding the Benjamini-Hochberg correction, which stands out from the Holm and Bonferroni correction, set the flag add\_bh to FALSE.

**Validation of simulation tool**

SambaR provides four functions to generate plots which allow users to validate the reliability of the simulation tool.

To generate a plot showing simulated (points) and expected (lines) proportion of retained variation directly after a bottleneck event, depending on minor allele frequency in the source population and the number of founders (ne), execute:

*runsim\_retained(n\_loci=1100,n\_selectedloci=1000,sel\_coef=0.1,sourcemafmean=c(0.01,0.1,0.4),ne=c(1,2,5),do\_heatmap=TRUE,do\_export=FALSE)*

Expected retained variation is described by the function: 1-((1-maf)2ne).

To generate a plot showing simulated (points) and expected (lines) fixation probabilities in founder populations, depending on the minor allele frequency (q) in the source population and the selection coefficient (s), execute:

*runsim\_fixation(n\_loci=1100,ne\_F=50,n\_selectedloci=1000,do\_export=FALSE,n\_gen=500)*

Expected fixation probability is described by the function:

*1-exp-2\*ne\_F\*q\*(s+0.0001)*

*1-exp-2\*ne\_F\*(s+0.0001)*

To generate a matrix with fixation probabilities as a function of founder population Ne and number of generations, execute:

*runsim\_fixationtime(n\_loci=1100,n\_selectedloci=1000,sel\_coef=0.1,maf\_source=0.15,do\_export=FALSE)*

The generated matrix can be compared to expectations from theoretical population genetics.

The average time to fixation is given by the function 2/s\*ln(2\*Ne). If the average fixation time given a certain combination of s and Ne is for example around 100 generations, then the fixation simulated probability after 500 generations should be close or equal to 1.

The starting allele frequency distribution of the source population, prior to the founder events, is generated by letting a uniform distribution (with fixed minor allele frequency) for a certain number of generations (default: 200 generations). To check if this burn-in time generates indeed a realistic MAF distribution, execute:

*multimafburnin(mymaf\_means=c(0.1,0.125,0.15,0.2),maf\_vector=NULL)*

For comparison, the user needs to provide to the flag maf\_vector an input vector with observed minor allele frequencies. Say for example that the source population in your true SNP dataset is called ‘Norway’, this could be: *snps$maf\_Norway[snps$filter].*

# 12. Solve or work around errors

Please put in some reasonable efforts of trouble-shooting before consulting me for help (which of course, if needed, I am happy to provide). This includes carefully reading the manual (including the red coloured ‘trouble shooting’ text boxes), and carefully reading the entire screen output, which might provide clues to what is causing the error. Often errors can be circumvented by omitting a particular analysis by setting a particular flag to FALSE, as will be suggested by the screen output.

Whenever one of the main functions runs into an error, rerun the function with the flag silent set to FALSE. For example:

*importdata(silent=FALSE)*

This will not solve the error, but allows to locate the error. If consulting me for help, please enclose the resulting entire screen output.

Here is another suggestion for a quick and dirty work around. It is not unlikely that encountered errors are related to non-essential function, for example plotting functions. Therefore a work around can be to disable the responsible the lines in the SambaR script, namely by putting a hashtag at the start of that line. How to know which lines to disable?

Say that you encounter an error whilst running:

*filterdata(silent=FALSE)*

Say furthermore that you get the following the screen output:

*plot\_indF*

*Error in plot.new() : figure margins too large*

The screen output seems to indicate that the error is related to the function plot\_indF(). Search within the SAMBAR script for the string:

*if(!silent){cat("plot\_indF",sep="\n")}*

Next, put hashtags in front of subsequent lines, until the next occurrence of a line which contains the string ‘*if(!silent)*’. For example:

*if(!silent){cat("plot\_indF",sep="\n")}*

*#plot\_indF(export="eps",plotname="Inbreeding")*

*#plot\_indF(export="pdf",plotname="Inbreeding")*

*#plot\_indF(export="png",plotname="Inbreeding")*

*if(!silent){cat("plotscatter\_indF",sep="\n")}*

Afterwards, reload the SambaR script (user the source function), and try rerunning the filterdata. Chances are that this time the function will complete without errors.

Note that, obviously, this solution only works for certain problems. If disabling vital parts of the process, SambaR can start working correctly altogether.