MANUAL

SambaR

Snp datA Management and Basic Analyses in R

MENNO DE JONG, MOLECULAR ECOLOGY LAB, DURHAM UNIVERSITY, UK

Latest update: 02-01-2019 *mennodejong1@hotmail.com*

SambaR is an 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 as far as I am aware, apart from a few options this package can also be used to analyse SNP datasets which are generated with other protocols. Sambar accepts both biallelic and multiallelic datasets, but at present it offers a very limited number of analyses for multiallelic datasets (and many for biallelic datasets).

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 all or most of your analyses (e.g. quality control, PCA, CA, DAPC, MDS, Nei's D, Fst, HWE, SFS, LD, theta, Tajima's D, 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. In that 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 automatically generated when running the importdata() function. This library can be exported into any referencing software, such as Endnote or Zotero.

The output

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 Distance. After executing all functions, these subdirectories will contain inputfiles, 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.

Although SambaR creates most plots in up to 4 formats (i.e. eps, png, pdf, and wmf) , I advise to use the pdf format. This format is vector based, meaning that zooming in won't make the picture pixular. In addition, pdf can deal with transparancy, and unlike wmf, the format does not cause problems when converting your word document to a pdf file. I focussed my time on getting the pdf files right, meaning that I might have overlooked issues with other formats.

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 will automatically be opened in a PDF viewer. Diagonal black lines will appear on the plot inserted in the Word document. If you close the plot in the PDF viewer, the diagonal black lines in the Word document disappear.

The default sizes of the labels, legends and symbols allow you to shrink most plots to just a few centimeters width and heigth, enabling you to create multitile plots using a minimum of space.

Content of this manual

Sommario

Content of this manual 3

1. Do all at once 4

2. Load SambaR..... 5

3. Load dependencies 6

4. Import your data into R..... 8

    4.1 Import biallelic data 9

    4.2 Import multiallelic data..... 16

5. Observe your data..... 17

6. Filter your data..... 19

7. Analyse your data 22

    7.1 Population structure 23

    7.2 Population differentiation 28

    7.3 Genetic diversity 30

    7.4 Selection analysis 33

    7.5 Association analyses 37

    7.6 Population demography (ADDITIONAL ANALYSIS) 38

8. Manage your data..... 43

9. Manage your plots 46

10. Understand your output 48

    10.1 The snps dataframe 49

    10.2 The inds dataframe..... 52

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')	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=9627819)	section 7.3
selectionanalyses(pairwise=TRUE)	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 only interested in doing selection analyses, 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 all run them 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'.

2. Load SambaR

To load SambaR into R, open an R session and load the source script, by typing on the R command line:

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.

Observe SambaR functions

To see the functions which are part of the SambaR package, type:

ls()

If you want to observe the script of a sambar function, type the name of the function without brackets. For example. if you want to see the script of the getpackages function, simply type:

getpackages   # so without brackets

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.

Another error you might encounter is:

Error in file(filename, "r", encoding = encoding) :

cannot open the connection

In addition: Warning message:

In file(filename, "r", encoding = encoding) :

cannot open file 'C:/ ProjectA/SNPanalysis/SAMBAR.txt': No such file or directory

If you get this error, you have most probably misspelled the path to the script. You can use the command 'list.files()' to locate the mistake. For example, to see which files are present in the directory specified above, you would type:

list.files("C:/ ProjectA/SNPanalysis/SAMBAR.txt")

3. Load dependencies

As mentioned before, SambaR depends heavily on other R packages. 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, 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 sources 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 might take a while to complete. Subsequent executions (for example after you restart the computer or after you restart R), will be much quicker.

To be sure everything worked fine, execute the getpackages function twice. If everything worked fine the first time, the second time all you should see on the screen is:
All required CRAN packages already installed.
All required Bioconductor packages already installed.
All required github packages already installed.
All R package versions stored in an dataframe called 'myversions'.

To check which packages are installed, type: *library()*
To check which packages are loaded, type: *(.packages())*
To see the version of each package, type: *myversions*
To export the citations of all packages used by SambaR, type: *getcitations()*

Trouble shooting

Getting dependencies is the most tricky part of R and therefore also of SambaR. Although ordinarily the getpackages()-function should work fine, many things could potentially go wrong when installing R packages and sometimes there is no straightforward solution.

In general I do not recommend to try to install the packages yourself, because you will likely use the same functions as the getpackages() function does for you automatically (e.g. install.packages() and library() in case of CRAN packages), so therefore you are likely to run into the same error anyway. If however the problem appears to be with one specific package (a missing dependency for example), you could of course always give it a try to install this package yourself and then rerun the getpackages() function.

Here is a list of things you can try (in ascending order of effort) before rerunning the getpackages() function:

1. Restart R.
2. Restart the computer.
3. Reinstall a different version of R.
   This only takes a few minutes and provides the best chance the problem will be solved. Even if you have a recent version installed, installing another version of R (both older or newer) might solve the problem, because all packages will be newly installed. Note that the version should be at least 3.5.
4. Update all packages.
   If you installed some packages before (using the same R version), it might also help to update all packages using the command: update.packages(ask=FALSE). This can take however ages and there is no guarantee it will solve the issue.

If neither of the above options provided a solution, you are free to contact me (but no guarantee I will be able to sort it out).

4. Import your data into R

If you have biallelic data (max two alleles per SNP), follow the instructions in section 4.1.
If you have multiallelic data (at least one SNP has more than two alleles), follow the instructions in section 4.2.

Is your data biallelic?
STACKS outputs biallelic data only, so if you generated your data using STACKs, this applies to you.
In general it goes that at present SambaR offers a limited amount of analyses for multi allelic data, so I recommend to input biallelic data.

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	PGDspider or vcftools
- Convert PED and MAP files to binary RAW and BIM files	PLINK
- 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
http://zzz.bwh.harvard.edu/plink/download.shtml

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.

Convert from vcf to PED/MAP using vcftools

Arguably the easiest way to create a PED and MAP format is to use vcftools, which requires a vcf file as input. I would 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:

```
/path/to/vcftools --vcf inputfile --plink --out prefixoutputfiles
```

One minor disadvantage of vcftools is 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
```

(There is a catch however: this command works only if the name

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 do 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 delimited file with no header and two columns. The first column contains the sample name and the second column contains the population name, like this:

```
Sample1	Pop1
Sample2	Pop1
Sample3	Pop2
Sample4	Pop2
```

etc.

WARNING: population names may NOT contain underscores or spaces, because this will interfere with Sambar commands. In addition, preferably your population names contain no more than 12 characters, as long names might not be (properly) displayed in some SambaR output plots. SambaR does not consider hierarchial structuring.

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, preferably your population names contain no more than 10 characters, as long names might not be (properly) displayed in some output plots. SambaR does not consider hierarchial structuring.

Convert from PED to RAW and MAP to BIM format
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 ‘recode A’ flag tells the software to create the raw file. The make-bed flag tells the software to make the bim file.

In order for this 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 prevents 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.

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 do, make sure to a plain text editor, such as Notepad or Notepad++, and not other programs such as Word, because this might corrupt the file.)

Trouble shooting

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

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)

These are the function arguments:

inputprefix:	prefix of RAW and BIM file (so without .raw and .bim extension)
sumstatsfile:	If TRUE (default is FALSE), the function expects to find within the working directory 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. If you generated your genotype data with the STACKS de novo pipeline (so without using a reference genome), this does not apply to you, and therefore leave this flag to FALSE. Otherwise, make sure that the name of the file is exactly 'sumstats.tsv', which means you have to edit the name, because STACKS provides a suffix. (SambaR uses the sumstatsfile to derive the positions of the snps in the reference genome. If your MAP file contains this information (i.e.: first and second column of the MAP file are not 0), then SambaR will derive this information from the MAP file.)
vcfsamplefile:	If the data is generated from an vcf-file using vcftools, the user needs to provide an additional tab delimited file with no header and two columns. The first column contains the sample name and the second column contains the population name. Provide the entire name of the file, including suffix. Default is NULL.
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.
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.
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, and I subsequently used vcftools to calculate read depths stored in files called out.idepth and out.ldepth.mean. I 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. If everything works fine, all which should be displayed on the screen are the following messages:

```
Reading PLINK raw format into a genlight object...
 Reading loci information...
 Reading and converting genotypes...
 Building final object...
...done.
Created two dataframes called 'snps' and 'inds', as well as a genlight object called 'mygenlight'.
Creating snps dataframe...
Creating inds dataframe...
Assigning population colours...
```

Extracting information from STACKS sumstats file...

Extracting read depth information...

Creating output directories...

Data succesfully imported.

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 before, 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 vcfsamplefile 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.

4.2 Import multiallelic data

SambaR uses the 'read.structure'-function of the Adegenet package to load multiallelic genotype data into R. This functions 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()
maples()        # correlation fis within pops vs fst between pops
meanr2()        # simulate expected correlation fis and fst
```

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 softwares and 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 equalling the number of loci, and the number of rows equalling 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 equalling the number of loci.
- A dataframe called '***inds***', containing sample specific information, with the number of rows equalling 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 unsure about the actual meanin 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)
To observe the snps file in R's built-in data editor, type:
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 this 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,chromosomes=FALSE,min_mac=2,dohefilter=TRUE)

Despite its name, the function does in general NOT remove any 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 indmiss and snpmiss flags in the filterdata()-function 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. Feel free to set these flags to any value you prefer. The filter settings you set now will be used for all subsequent analyses. (But if later on you decide you want other settings, you can quickly and easily rerun the analyses with different filter settings.)

The min_mac flag 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 SNPwhich 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 loci which are strongly out of Hardy-Weinberg-Equilibrium. These loci are highlighted in red in the He_vs_maf plot which is generated by the filterdata function.

The output
Once the execution 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$filter2 (non thinned dataset). These columns determine whether samples and loci will or will not be used during subsequent analyses. If TRUE, the sample or locus will be included in the analysis. If FALSE, the sample or locus won't be considered. For population structure and genetic diversity analyses, SambaR functions will use the thinned dataset (snps$filter). This is to avoid issues related to linkage disequilibrium. For selection analyses, SambaR will use the non thinned dataset (snps$filter2).
To avoid errors during the execution of subsequent functions, NEVER remove or rename the inds$filter, snps$filter and snps$filter2 columns.

Trouble shooting
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)

How to change filter settings

Although the snps$filter, snps$filter2 and inds$filter columns should never be removed or renamed, you are allowed to edit them, at any time. The most straightforward way is rerun the filters()-functions with the desired settings.

If you want to be fast, and if you have basic settings, you can also run single line commands. For example, if you want to exclude samples with more than 5 percent missing data, you type:

inds$filter <- inds$miss<=0.05

If you want to exclude snps with more than 2 percent missing data and/or with a minor allele frequency below 0.05, you type:

snps$filter <- snps$miss<=0.02&snps$maf>=0.05

Additional analysis: distance per chrom

If you called snps using reference mapping (rather than denovo), and if your reference genome is assembled to chromosomes (rather than to scaffolds), set chromosomes flag to true:

filterdata(indmiss=0.25,snpmiss=0.10,chromosomes=TRUE)

Next to the the standard tables and plots, the function will return a boxplot and table distance between adjacent snps per chromosome. If set to TRUE but chromosome numbers are absent, the following error will occur:

Error in do.call(rbind, tapply(snps$dist[snps$dist2 & snps$placed], snps$chr[snps$dist2 & : second argument must be a list

You can ignore the following warning message:

Warning message: In importdata: NAs introduced by coercion

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, you need to generate input files. To do so, execute the following function once the filterdata() function has finished:

exportsambarfiles()

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$filter column
- and a dataframe called 'snps', containing an snps$filter' and snps$filter2 column.

You created these objects in the previous steps.

Trouble shooting

If any of the objects listed above are missing – for example because you didn't run the filterdata() command yet, or because you renamed some columns afterwards – SambaR functions will return errors.

If any of the columns in the inds or snps dataframe contain a NaN value (infinite value), you might encounter the following error when generating a plot:

Error in plot.window(...) : need finite 'ylim' values

Say the NaN value occurs in the column inds$miss, you can work around this by typing:

inds$miss[!is.finite(inds$miss)] <- 1

If directories have been deleted, you might get an error like:

Error in setwd(sambardir) : cannot change working directory

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)
In which Kmax determines maximum number of clusters to be considered.

This function will produce a lot of rubbish on the screen, and in the meantime generate:

- Fis plot showing (Hexp-Hobs)/Hexp estimates per population. Deviation from 0 is indicative of population substructure.
- a Nei genetic distance matrix (between individuals) generated by software StAMPP
- PCoA (Principal Coordinates Analysis) plots generated by R package Ape, based on Nei's genetic distance (StAMPP), Hamming's genetic distance (poppr) and theta (SambaR)
- PCA (Principal Component Analysis) plots generated by R package SNPrelate
- multidimensional scaling (MDS) plots generated with R functions dist, cmdscale (metric MDS) and isoMDS (non-metric MDS), the latter function from R package MASS
- DAPC scatter plots for various K (default is 2-6) generated by Adegenet
- matrices with expected clusters vs DAPC inferred clusters for various K (default is 2-5)
- correspondence analysis (ca) plots, generated with R package factoMiner
- barplots with Bayesian population assignment probabilities (if more than 1 pop), showing likelihood that a sample belongs to either of the predefined populations
- genotype networks (if more than 1 pop), based on 1000 randomly randomly snps
- structure plots with ancestry coefficients generated by software LEA
- optionally (if columns longitude and latitude are present in inds dataframe): geographical maps with sample piecharts representing admixture analyses qmatrices, both for LEA output and DAPC output, and both for the metapopulation and for each population separately
- optionally (if columns longitude and latitude are present in inds dataframe): maps with piecharts, and mantelplots showing geographical distance vs genetic distance (theta), stored in the 'Maps' subdirectory

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 tell which samples clusters together. Depending on the similarity of your samples, you might find that a few individuals stand out, and that all other individuals are clumped together. If so, you might want to run the findstructure analysis again, this time excluding the individuals which stand out. To do for example the analysis again without individuals numbered 6, 20, and 43, type:

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

findstructure()

Don't forget to afterwards include these samples again for subsequent analyses:

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

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)

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 genetic distance calculations:

findstructure(do_nei=FALSE)

Occassionally, you migh 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)

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) vcfsamplefile as explained elsewhere in this manual. Store this file in the directory with your input PED and MAP files, and type:

replacepop(samplefile="filename.txt")

Another way is to reassign samples to populations is to make changes in the first column of the original PED file (which defines the population names), rerun PLINK to convert PED/MAP to bed/raw files, and afterwards rerun the importdata() function.

Don't forget to rerun the filterdata() function.

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.

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)

ADDITIONAL ANALYSIS: TESS3r

SambaR provides functions to run spatial genetic structure analyses, using the R packages TESS3r and Geneland. Because both programs take a while to run, especially Geneland, the user has to call these functions separately. For both functions SambaR expects to find within the inds dataframe columns named 'longitude' and 'latitude' (included during execution of the importdata() function).

To run and plot the output of Tess3r, type:

gettess()

runandplottess3(coordfile,export="pdf")

ADDITIONAL ANALYSIS: Geneland

SambaR provides functions to run spatial genetic structure analyses, using the R packages TESS3r and Geneland. Because both programs take a while to run, especially Geneland, the user has to call these functions separately. For both functions SambaR expects to find within the inds dataframe columns named 'longitude' and 'latitude' (included during execution of the importdata() function).

Geneland needs to be installed from either a zipfile (Windows) or tarball (Unix). You can download these files from: https://i-pri.org/special/Biostatistics/Software/Geneland/distrib/

To install on a Windows computer, type on R command line:

getgeneland(mypath="C:/path/to/Geneland_4.0.8.zip",mylib=NULL)

To install on a Unix computer, type on R command line:

getgeneland(mypath="/path/to/Geneland_4.0.8.tar.gz",mylib="/path/to/your/storage/dir/")

To run Geneland and subsequently plot the output, type:

runandplotgeneland(my_iter=10000,burnin_fraction=0.2,my_thin=100,nloci=1000)

The settings my_iter, burnin_fraction, and my_thin can be used the alter the MCMC chain settings. The flag nloci determines how many (randomly selected) loci will be used for the analysis. This number can, obviously, not exceed the number of retained snps in your snps dataframe. Depending on the capacities of your computer, there are limits to the combinations of my_iter, my_thin and nloci. If you set these values too high, you will encounter an error about memory allocation. Therefore you might not be able to use all your snps.

The output will be stored in the folder 'Genelandoutput' in the structure directory.

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()
```

Note: it only makes sense to run this command if your dataset contains more than 1 population. If you have only 1 population, Sambar will not perform any calculations.

Screen output:

```
Calculating Nei's genetic distance...
Calculating pairwise Wright Fst...
Calculating pairwise Weir & Cockerham Fst...
Calculating pairwise Weir & Cockerham 1983 multilocus Fst...
Outputting summary table...
Analysis finished.
```

This function will generate:

- a Nei genetic distance matrix (between populations) generated by software StAMPP
- a heatmap with multilocus Weir & Cockerham Fst
- a table with pairwise population comparison Fst estimates
- Heterozygosity vs Fst scatterplots

The function will also add columns to the snps dataset with singlelocus pairwise population Wright Fst estimates and Weir & Cockerham 1983 Fst estimates.

ADDITIONAL ANALYSIS: Plot PLINK relatedness (kinship) estimates

SambaR contains a function to plot sample relatedness measures generated by PLINK.
For that purpose, you will find, if you have run the exportsamberfiles() function, within the inputfiles directory two files called 'metapop.filter.number.ped' and 'metapop.filter.number.map'.

Navigate on the command line of your computer to this directory and execute this plink command:
/path/to/plink --file metapop.filter.number –cow –-allow-extra-chr --genome

This will output to the Sambar_output directory a file called 'plink.genome'.
In R, execute:
plotrelatedness(export="pdf")

This function will export three plots to the Divergence subdirectory:

- Relatedness.between.pdf
- Relatedness.within.pdf
- Relatedness.persample.pdf

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.

7.3 Genetic diversity

You might also want to know how much genetic variation each population contains.
To find out, you can execute the following command:
calcdiversity()

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 sumstats_summary.tsv file (i.e. batch_1.sumstats_summary.tsv or populations.sumstats_summary.tsv, depending on your version of STACKS), which is outputted by the STACKS populations command. The total number of sites is listed in the third column ('Variant sites') of the second part of this file, after the line '# All positions (variant and fixed)'. You have a value for each population, and those values might differ from each other. In that case choose the maximum value.

If for example the combined length of your loci is 9627819 base pairs, the command would be:
calcdiversity(nrsites= 9627819)

If you provide a value to the nrsites flag, the function will generate:

- scatterplot with genome wide heterozygosity vs nucleotide diversity
- scatterplot with theta and Watterson's theta estimates, and inset with barplot showing Tajima's D estimates
- histograms with locus specific HWE chisquared test scores
- a table with summary statistics
- 2D folded SFS plots (for each pairwise population comparison)
- folded (and binned) SFS line plot
- SFS vector for each population
- violin plots of genome wide heterozygosity per population

All you should see on the screen is:
Generating site frequency spectra...
Calculating theta and tajima's D per population...
Plotting genome wide heterozygosity and theta...
Generating summary statistics...
Analysis finished.

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 1000000 --ld-window-r2 0 –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 Diversity 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.

ADDITIONAL ANALYSIS: Runs of homozygosity

If you have positional information about your snps (so if the dataset is generated using a reference genome rather than generated denovo), and if your reference genome is assembled up to chromosomes (rather than contigs or scaffolds), you might also want to execute for various chromosomes:

multiglPlots(chromosome=1,export="pdf")

This function will export a colourful plot called 'ROH.pdf' to the Diversity directory. White indicates missing data. Black indicates homozygous major allele. Grey indicates homozygous minor allele. Colours indicate heterozygous sites. Therefore, long stretches with no colours indicate runs of homozygosity (ROH).

7.4 Selection analysis

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)

To execute each method, type:

selectionanalyses(do_meta=TRUE ,export="pdf",do_fsthet=FALSE)
selectionanalyses(do_meta=FALSE,do_pairwise=TRUE,export="pdf" do_fsthet=FALSE)
selectionanalyses(do_meta=FALSE,do_pheno=TRUE,export="pdf",phenolabels=c("high","low"))

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.

The output

Identified outliers are possibly 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, often not unlikely, false positives. Have a look at the allele frequencies of the outlier SNPs and consider whether the allele frequencies are different or similar enough between/among populations to justify the influence of selection.

SambaR's GWDS test implements the exact fisher test to search for correlations between trait and minor allele frequency, and subsequently compares locus specific scores to the overall mean scores to search for outliers. This function returns a Manhattan plot 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.

Trouble shooting

You might run into the following errors:

Calculating FSTs, may take a few minutes...

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

or:

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

These errors are related to OutFlank and unfortunately I don't know how to fix them. The only solution I can offer 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: 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: Sliding window Fst

If your reference genome is assembled to chromosomes (rather than contigs or scaffolds), you can do a sliding window Fst analysis. In the example of the marine and the freshwater populations you might for example be looking for overlapping peaks of Fst values of pairwise comparisons between marine and freshwater populations. Overlapping peaks could occur by chance, but they could also be indicative of convergent adaptation.

First select which pairwise population comparisons you want to execute. To find out, type:
combn(populations,m=2)
This will print a table on the screen. Find the column numbers of the pairwise population comparisons you are interested in.

Next generated the sliding window fst analysis. Say that you are interested in pairwise comparisons 2 and 3, type:
plotwindowfst(export="pdf",mycomparisons=c(2,3),addX=TRUE,myyrange=c(0.025,0.15))
This will function will export pdf files to the 'Selection'-directory.

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:
createbayescaninput(allpairwise=FALSE)

To execute PGDspider and Bayescan, type on command line:
module add java
java -jar PGDSpider2-cli.jar -inputfile pheno.filter2.ped –outputfile pheno.filter2.geste -spid PED2Bayescan.spid &
bayescan_2.1 pheno.filter2.letter.geste -od bayescan_pheno -threads 3 &

To subsequently incorporate Bayescan output in Sambar plots, type:
selectionanalyses(do_meta=TRUE,addbayescan=TRUE)

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.

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 (because 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. (In other words: STACKS user can derive this value from the sumstats.summary.tsv file. 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: 9627819 # 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)

The SFS is a vector which shows the number of SNPs which have a certain 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 consists of 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 represent 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, 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

Occassionaly 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.

10. Understand your output

This section contains descriptions of the columns of the snps (10.1) and inds (10.2) dataframe.

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 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)