`diveRsity` v1.3.2 Help Manual

(compiled version)

by *Kevin Keenan*

kkeenan02@qub.ac.uk

`http://diversityinlife.weebly.com/`

November 8, 2012

Contents

1 Introduction

This manual has been written as a more generic, user-friendly guide to using diveRsity in the R environment than the help PDF distributed with the package on CRAN. It will outline briefly how to get the latest version of R, how to install the diveRsity package as well as how to install suggested packages. Fully reproducible Worked examples for functions will be provide as a guide to how the package should be implemented. Effort has been made to keep R jargon to a minimum to ensure accessibility for R beginners.

1.1 About R

R is an extremely powerful and popular software for statistical programming. It is very well supported by a dedicated group of people known as the *R core development team* [1], as well as an active community of developers and useRs. More information about R can be found at http://www.r-project.org/about.html.

1.2 About diveRsity

diveRsity is a package containing five functions written in the statistical programming environment R. It allows the calculation of both genetic diversity partition statistics (e.g. G_{ST} & F_{ST}), genetic differentiation statistics (e.g. G'_{ST} and D_{Jost}), and locus informativeness for ancestry assignment (e.g. I_n), as well as basic population parameters such as allele frequencies. diveRsity provides useRs with various option to calculate bootstrapped 95% ci's both across loci and for pairwise population comparisons. All of these results are returned in convenient formats and can be plotted interactively.
The calculation of diversity statistics such as G_{ST}, G'_{ST} and D_{est} is carried out using the function div.part, locus informativeness for ancestry inference (i.e. I_n) is calculated using in.calc and basic population statistics are calculated using readGenepop.user. Full descriptions and explanation of functions are provided below.
diveRsity was written to ensure that even R beginners can carry out genetic analyses in R without major difficulties. By automatically writing analysis results to file, useRs do not need to understand how to access variables in the R environment, let alone know what a variable is. However, for more experienced useRs, all analysis functions return results variables to the R environment, details of which are provided in the **Function usage** section below.

1.2.1 What's new?

Versions 1.2.0 and up introduce a complete rewrite of `diveRsity v1.0`. All subsequent versions have been vectorized in all but the least computationally intensive pieces of code, resulting in much faster execution speed.

Parallel computations are also now available when using the `in.calc` and `div.part` functions. These two major changes mostly affect the speed at which the program executes. An additional results object, (i.e. `pairwise`) is now also returned from the function `div.part`. This additional functionality now allows useRs to calculate pairwise statistics without having to run the computationally intensive bootstrap algorithm, thus saving time.

As of version 1.2.3, Weir and Cockerham's (1984) F-statistics are also calculated for global estimates, locus estimates and pairwise population estimates as well as 95% confidence intervals in the function `div.part`.

The calculation of Weir and Cockerham's F-statistics increases analysis time by around 0.3 seconds per bootstrap replicate, thus leading to significant increases in overall exacution time if a large number of bootstrap iterations are used. For this reason, the calculation of F-statistics has been included as an optional extra through the new argument `WC_Fst`.

Versions 1.3.0 an up includes additional plotting functions to aid in data visualisation. These new functions are;

`corPlot` - provides useRs with the ability to plot locus G_{ST}, θ, G'_{ST} and D_{Jost} against the number of alleles at each locus. This method may be useful to assess whether particular loci might be suitable for the inference of demographic processes (i.e. they are not unduly affected by mutation).

`difPlot` - is a function intended to be used as a data exploration tool. This function plots pairwise estimated statistics, allowing useRs to easily visualise pairwise comparisons of interest (e.g. highly differentiated population pairs).

Version 1.3.2 provides a more flexibility in reading genepop files. It also returns more informative error when genepop files are in the wrong format. This version also fixes a bug in writing results to disk. If `outfile` is set to `NULL` in the functions `div.part` or `in.calc`, no directories will be created.

2 Setup

2.1 Installing `R`

To use `diveRsity` you will need to download and install `R`.
It is available at:
`http://cran.r-project.org/`
Simply download the R distribution appropriate for your operating system and install as normal.

2.2 Installing `diveRsity`

`diveRsity` is currently available on `CRAN` (The Comprehensive R Archive Network), thus installation is simple. Launch `R`, and in the console (you will see the '`>`' symbol when `R` is ready for you to type), use the following command:

```
install.packages("diveRsity")
```

The package will be updated regularly, both with added functionality and to fix bugs. The most up to date notes and versions of the package can be found at:
`http://diversityinlife.weebly.com/software.html`.

2.3 Installing optional enhancer packages

The dependency `plotrix` [2] will download automatically if you install `diveRsity` from `CRAN`. Suggested/optional packages must be installed manually (excluding `parallel`, which is distributed with `R`.
Optional packages are:
`xlsx` — writes results to .xlsx. [3]
`sendplot` — Plots results to .html files with tool-tip information. [4]
`doSNOW` — Used in parallel computations (Linux). [5]
`doParallel` — Used in parallel computations (Windows). [6]
`snow` — Used in parallel computations (Linux). [7]
`parallel` — Used in parallel computations (Linux & Windows). [8]
`foreach` — Used in parallel computations (Linux & Windows). [9]
`iterators` — Used in parallel computations (Linux & Windows). [10]

Each of these packages can be installed using the below command;

```
install.packages("package_name")
```

Just replace ‘package_name’ with the name of the package you want to install. See ?install.packages for details.

2.4 Loading diveRsity

To load diveRsity in the current R session, type the following into the R console:

```
library("diveRsity")
```

You will not need to load any of the other dependencies or optional packages as diveRsity will do this as and when it uses additional packages. After loading diveRsity into your current R session all of its functions are available for you to use.

For convenient access to usage information on each function type:

```
? div.part
? in.calc
? readGenepop.user
? corPlot
? difPlot
```

Each of these commands will provide information on function usage. The help pages associated with each function describe in detail how each argument should be passed to the function.

3 Function details

3.1 `div.part()`

`div.part` (diversity partition), allows for the calculation of three main diversity partition statistics and their respective estimators. The function can be used to mainly explore locus values to identify 'outliers' and also to visualise pairwise differentiation between populations. Bootstrapped confidence intervals are calculated also. Results can be optionally plotted for data exploration purposes. The statistics and their basic formulae are as follows:

3.1.1 Standard formulae

G_{ST} **[11, 12]**

$$G_{ST} = \frac{D_{ST}}{H_T} \tag{1}$$

Where $D_{ST} = H_T - H_S$, H_T is the total heterozygosity and H_S is intra-population heterozygosity.

G'_{ST} **[13]**

$$G'_{ST} = \frac{G_{ST}}{G_{ST(max)}} \tag{2}$$

Where G_{ST} is as above, $G_{ST(max)} = \frac{H_{T(max)} - H_S}{H_{T(max)}}$ and $H_{T(max)}$ calculated as $H_{T(max)} = \frac{(k-1+H_S)}{k}$ and is the maximum possible H_T value given the observed within sample heterozygosity.

D_{Jost} [14]

$$D_{Jost} = \left[\frac{(H_T - H_S)}{(1 - H_S)}\right]\left[\frac{n}{(n-1)}\right] \tag{3}$$

Where H_T and H_S are as defined above, and n is the number of population samples.

3.1.2 Estimator formulae

The estimators of both G_{ST} and G'_{ST} were calculated by simply substituting the H_S and H_T components of each statistic with their estimators calculated using equations 4 and 5 respectively. $D_{estChao}$ was calculated using the method described in [15] (eqn 6 below). The formulae are as follows:

$\hat{H}_S$ **[12]**

$$\hat{H}_S = H_S \left[\frac{2\bar{N}}{(2\bar{N} - 1)} \right] \tag{4}$$

Where H_S is the inter-population heterozygosity and $\bar{N}$ is the harmonic mean of sample size across all samples.

$\hat{H}_T$ **[12]**

$$\hat{H}_T = H_T + \left[\frac{\hat{H}_S}{(2\bar{N}n)} \right] \tag{5}$$

Where H_T is the total heterozygosity, $\hat{H}_S$ is as defined in equation (4), $\bar{N}$ is the harmonic mean of sample sizes and n is the number of population samples.

$D_{est(Chao)}$ [15, 14]

$$D_{est(Chao)} = \frac{1}{[(\frac{1}{A}) + var(D)(\frac{1}{A})^3]} \tag{6}$$

Where A is the arithmetic mean of D_{Jost} across loci, and $var(D)$ is the variance of D_{Jost} across loci.

F_{ST} (i.e. $\hat{\theta}$) [16, 17]

$$\hat{\theta} = \frac{\hat{\sigma}_P^2}{\hat{\sigma}_P^2 + \hat{\sigma}_I^2 + \hat{\sigma}_G^2} \tag{7}$$

Where $\hat{\sigma}_P^2$ is the sum of variance components for populations, $\hat{\sigma}_I^2$ is the sum of variance components for individuals within populations and $\hat{\sigma}_G^2$ is the sum of variance components for alleles within individuals.

3.1.3 Bootstrapping

The variance each statistic can be assessed using the bootstrapping method implemented in `diveRsity`. 95% confidence intervals are calculated using the method described in [18].

3.2 `in.calc()`

`in.calc` allows the calculation of locus informativeness for the inference of ancestry both across all population samples and pairwise comparisons. These parameters can be bootstrapped using the same procedure as above to obtain 95% confidence intervals. The basic equations for both the allele specific and locus specific calculation of I_n are as follows:

$I_n(alleles)$ **[19]**

$$I_n(Q; J = j) = -p_j log_e p_j + \sum_{i=1}^{K} \frac{p_{ij}}{K} log_e p_{ij} \tag{8}$$

Where p_j is the parametric mean frequency of the j^{th} allele across populations, log_e is the natural logarithm, p_{ij} is the frequency of the j^{th} allele in the i^{th} population, and K is the number of populations.

$I_n(locus)$ **[19]**

$$I_n(Q; J) = \sum_{j=1}^{N} I_n(Q; J = j) \tag{9}$$

Where N is the number of allele at the locus of interest and $I_n(Q; J = j)$ is as in equation 7.

3.3 readGenepop.user()

Although the `readGenepop.user` function is used extensively in both `div.part` and `in.calc`, its complexity is well hidden from general useRs. However, it has been included in `diveRsity` as a usable function for more experienced useRs, who may find it useful for data exploration and the development of analysis methods. As of version 1.3.0, this function is also implemented for use with the function `corPlot`. The function `readGenepop.user` returns up to 18 distinct `variables` (described in detail below), some of which have particularly complex structures. Although this manual provides basic summaries of each returned `variable`, for the function to be useful, useRs are advised to explore the individual objects. This can be done using functions such as `str`, `names` and `typeof`.

3.4 corPlot()

New to v1.3.0
This function allows useRs to graphically visualise the relationship between locus polymorphism (i.e. Number of alleles) and corresponding G_{ST}, θ, G'_{ST} and D_{Jost} values per locus. This information is plotted along with the respective Pearson's product-moment correlation coefficients for each comparison. This information is intended to help useRs to decide whether it would be appropriate to use their particular loci for the inference of demographic processes (i.e.effective number of migrants per generation). Typically this is done following the relationship between F_{ST} and Nm arising under the finite-island model from the following formula:

$$F_{ST} \approx \frac{1}{4Nm + 1} \tag{10}$$

Where F_{ST} is the standardised measure of genetic variance among populations (i.e. G_{ST} or θ in this package), N is the effective number of breeding individuals and m is the migration rate among populations.

The requirement to validate the use of certain marker types to infer demography is particularly important given that such information is often used to inform conservation and management strategies. It has been shown extensively in the literature that the relationship between F_{ST} and Nm in equation 10 breaks down if other evolutionary forces are strong (e.g. [20]). For example if migration rate (m) is not $\gg$ than mutation rate (μ), then $F_{ST} \neq \frac{1}{4Nm+1}$, and the quantity Nm cannot be accurately derived.

For marker loci to be useful in the inference of demography, the the effects of such demographic processes must be detectable independently of the affects of processes such as mutation. As demographic processes are expected to have similar effects at all neutral loci, it is reasonable to expect that where mutation/selection/range constraints are having negligible effects on divergence at a particular set of loci, F_{ST} should be more or less equal across these loci. `corPlot` allows useRs to visualise if this is in fact the case. In general, the function will allow useRs to determine if mutation (assumed to be a major factor contributing to the the number of allele per locus), is having a noticeable effect on F_{ST} thus rendering them unsuitable for the inference of demography. As `corPlot` returns correlation plots of G_{ST}, θ, G'_{ST} and D_{Jost} against the number of alleles per locus, useRs have the additional benefit of assessing the effect of mutation on the differentiation statistics (e.g. D_{Jost}), which are more sensitive to the effects of mutation.

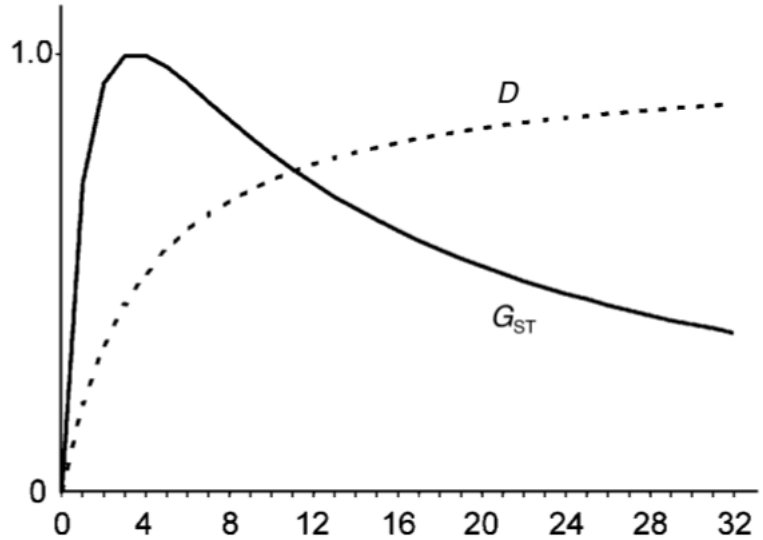
There is both theoretical and empirical evidence for this approach to assessing of the effects of processes other than migration and drift on divergence at neutral loci. O'Reilly *et al* (2004) [21], demonstrated from empirical data that F_{ST} (i.e. θ, specifically) had an inverse relationship with allelic richness in walleye pollock. Although the authors of this study attributed this observation to homoplasious mutations, the general affect is the same (i.e. mutational processes obscure divergence due to demographic processes). The results from this study can also be interpreted in light of the fact that F_{ST} has a theoretical maximum value defined as:

$$F_{st(max)} = \frac{H_{T(max)} - H_S}{H_{T(max)}} \tag{11}$$

Where $H_{T(max)}$ is the maximum possible total heterozygosity given the observed subpopulation heterozygosity, H_S.

Thus, because of the negative dependence of F_{ST} on heterozygosity, and the positive dependence of heterozygosity on number of alleles, we can predict a negative relationship between F_{ST} and number of alleles. The thrust of this argument is depicted in the figure below, where we can see the response of both G_{ST} and D_{Jost} to the number of unique alleles at a locus (Following Jost 2008 [14]).

The relationship between the number of unique alleles per subpopulation and G_{ST} and D_{Jost}



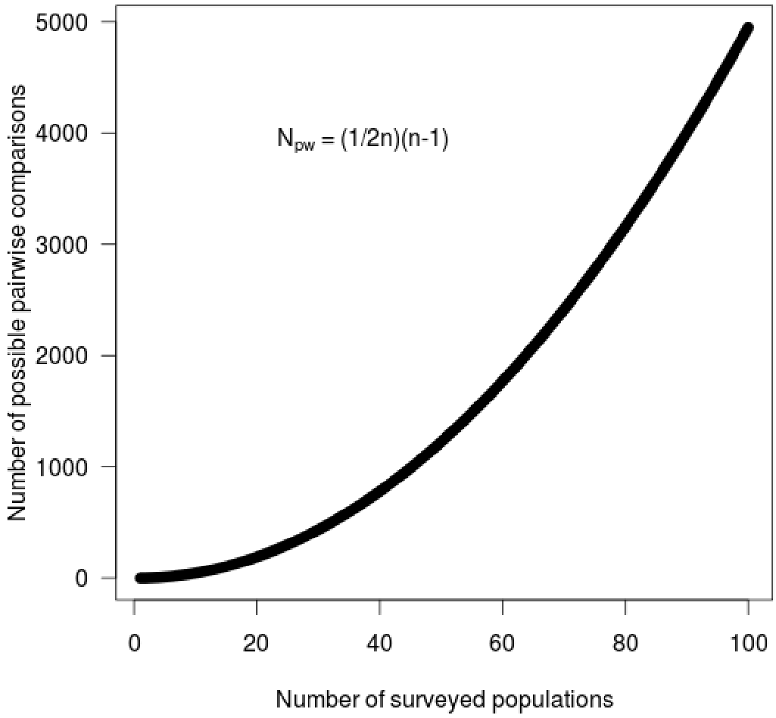
From this figure, it is clear that where the number of alleles at a locus is high (thus heterozygosity is high), G_{ST} is expected to be low. It is important to note that the negative relationship for G_{ST} and positive relationship for D_{Jost}, are complex, with multiple contributing factors. Thus this method should not be seen as definitive, but rather as a method to assess whether caution must be exercised when applying a particular marker set to address specific questions in populations of interest.

3.5 `difPlot()`

New to v1.3.0

`difPlot` is yet another plotting function introduced to help useRs easily visualise trends in their analysis results. Modern population genetic studies typically involve large numbers of population samples. It is often useful to know the pairwise relationships between each of these population samples. Due to the relationship between the number of sampled populations and the maximum number of possible pairwise comparisons, shown below, pin-pointing comparisons of interest can be very difficult.

The number of possible pairwise comparisons as a function of the number of sampled populations



To overcome this problem, `difPlot` plots the pairwise values calculated by the function `div.part` using a diagonal matrix coupled with a colour gradient used to indicate the magnitude of a particular pairwise value. The function plots the estimated pairwise values for G_{ST}, θ, G'_{ST} and D_{Jost}.

4 Function Usage

In this section the arguments and returned values for each function are explained.

4.1 div.part()

The general usage of this function is as follows:

```
div.part(infile, outfile = NULL, gp = 3, WC_Fst = FALSE,
         bs_locus = FALSE, bs_pairwise = FALSE,
         bootstraps = 0, Plot = FALSE, parallel = FALSE)
```

4.1.1 Arguments

`infile` Specifies the name of the *‘genepop’* [22] file from which the statistics are to be calculated This file can be in either the 3 digit of 2 digit format. The name must be a character string.

`outfile` Allows useRs to specify a prefix for an output folder. Name must a character string enclosed in either “” or ‘’.

`gp` Specifies the digit format of the `infile`. Either 3 (default) or 2.

`WC_Fst` A logical indication as to whether Weir and Cockerham’s, 1984 F-statistics should be calculated. This option will increase analysis time.

`bs_locus` Gives useRs the option to bootstrap locus statistics. Results will be written to *.xlsx* workbook by default if the package `xlsx` is installed, and to a *.html* file if `Plot=TRUE`. If `xlsx` is not installed, results will be written to *.txt* files.

`bs_pairwise` Gives useRs the option to bootstrap statistics across all loci for each pairwise population comparison. Results will be written to a *.xlsx* file by default if the package `xlsx` is installed, and to a *.html* file if `Plot=TRUE`. If `xlsx` is not installed, results will be written to *.txt* files.

Arguments cont.

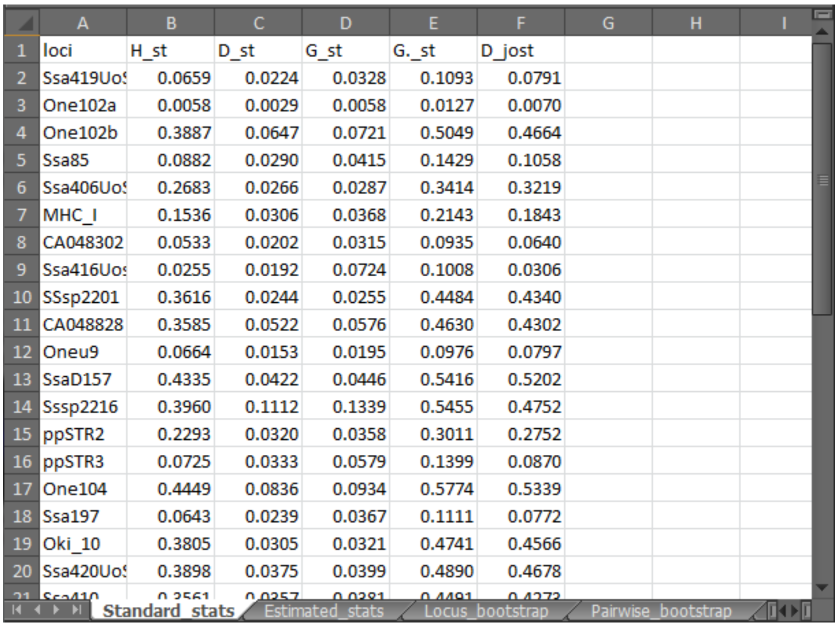
`bootstraps` Determines the number of bootstrap iterations to be carried out. The default value is `bootstraps = 0`, this is only valid when all bootstrap options are false. There is no upper limit on the number of bootstrap iterations, however very large numbers of bootstrap iterations for pairwise calculations (> 1000) may take a long time to run for large data sets and may also lead to excessive RAM consumption. As an example, a test data set containing over 4000 individuals across 97 population samples typed for 15 microsatellite loci, took 1.5 days to complete on a Windows 7 ultimate 64bit machine with an Intel Core i5-2435M CPU @ 2.40GHz x 4.

`Plot` Optional interactive *.html* image files of the plotted bootstrap results for loci if `bs_locus = TRUE` and pairwise population comparisons if `bs_pairwise = TRUE` and the package `sendplot` is installed. The default option is `Plot = FALSE`.

`parallel` A logical argument specifying if computations should be run in parallel on all available CPU cores. If `parallel = TRUE`, batches of jobs will be distributed to all cores resulting in faster completion. In Windows, the packages `doParallel`, `iterators`, `parallel` (distributed with `R`) and `foreach` must be installed to use parallel computation. In Linux the packages `doSNOW`, `parallel`, `snow`, `iterators` and `foreach` should be installed.

4.1.2 Returned values

Results returned by `div.part` vary depending on the argument options chosen. If the packages `xlsx` and `sendplot` are installed, results will be written to a single *.xlsx* workbook and *.png*/*.html* files providing `Plot = TRUE`. Alternatively, if these packages are unavailable the plot option is no longer available. Results will be written to multiple *.txt* files, the number of which varies between three and five depending on the argument options chosen. An example screenshot of the *.xlsx* output file is shown below:

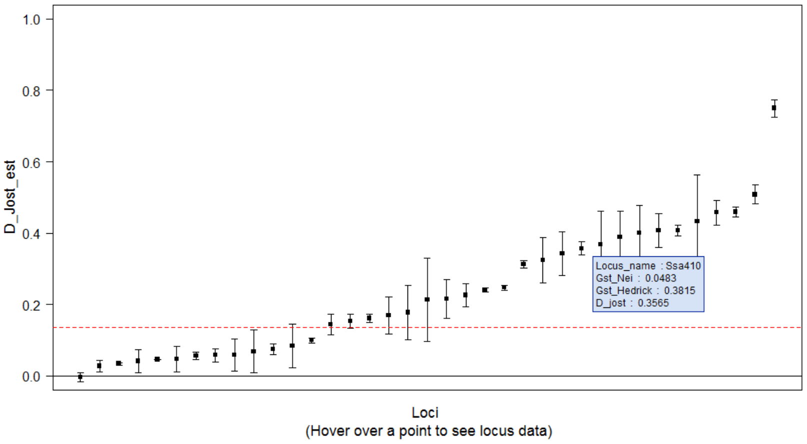


	A	B	C	D	E	F	G	H	I
1	loci	H_st	D_st	G_st	G._st	D_jost			
2	Ssa419UoS	0.0659	0.0224	0.0328	0.1093	0.0791			
3	One102a	0.0058	0.0029	0.0058	0.0127	0.0070			
4	One102b	0.3887	0.0647	0.0721	0.5049	0.4664			
5	Ssa85	0.0882	0.0290	0.0415	0.1429	0.1058			
6	Ssa406UoS	0.2683	0.0266	0.0287	0.3414	0.3219			
7	MHC_I	0.1536	0.0306	0.0368	0.2143	0.1843			
8	CA048302	0.0533	0.0202	0.0315	0.0935	0.0640			
9	Ssa416UoS	0.0255	0.0192	0.0724	0.1008	0.0306			
10	SSsp2201	0.3616	0.0244	0.0255	0.4484	0.4340			
11	CA048828	0.3585	0.0522	0.0576	0.4630	0.4302			
12	Oneu9	0.0664	0.0153	0.0195	0.0976	0.0797			
13	SsaD157	0.4335	0.0422	0.0446	0.5416	0.5202			
14	Sssp2216	0.3960	0.1112	0.1339	0.5455	0.4752			
15	ppSTR2	0.2293	0.0320	0.0358	0.3011	0.2752			
16	ppSTR3	0.0725	0.0333	0.0579	0.1399	0.0870			
17	One104	0.4449	0.0836	0.0934	0.5774	0.5339			
18	Ssa197	0.0643	0.0239	0.0367	0.1111	0.0772			
19	Oki_10	0.3805	0.0305	0.0321	0.4741	0.4566			
20	Ssa420UoS	0.3898	0.0375	0.0399	0.4890	0.4678			
21	Ssa410	0.3561	0.0357	0.0381	0.4491	0.4273			

Returned values cont.

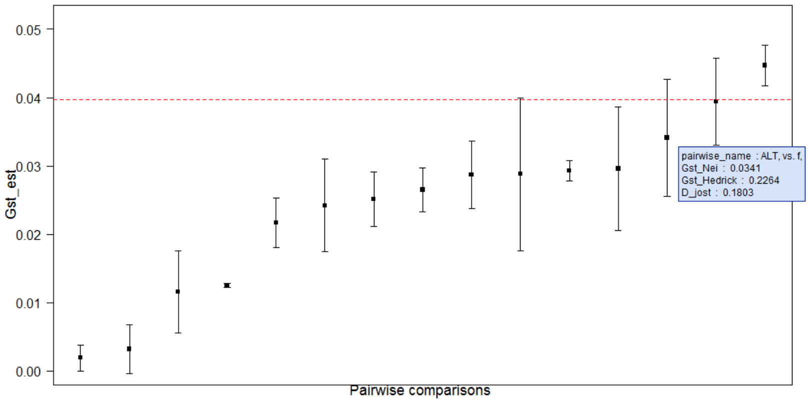
Examples of the interactive plots written, if `xlsx` is available, are given below. Error bars represent bootstrapped 95% confidence intervals, and the red dotted lines represent the global statistic values.

Example of bootstrapped locus results plot



Example of bootstrapped

pairwise results plot



pairwise_name : ALT, vs. f,
Gst_Nei : 0.0341
Gst_Hedrick : 0.2264
D_jost : 0.1803

(Hover over a point to see pairwise data)

Returned values cont.

For useRs wishing to carry out post analysis manipulations, all results from `div.part` are returned to the `R` environment. Depending on the bootstrap options chosen these results include between three to five of the `variables` below:

`$standard` A matrix containing identical data to the *Standard_stats* worksheet in the `.xlsx` workbook or the *Standard-stats[div.part].txt* text file. The last row in this matrix represents statistics calculate across all population samples and loci.

```
          H_st   D_st   G_st G_hed_st D_jost
Locus1  0.0659 0.0224 0.0328   0.1093 0.0791
Locus2  0.0058 0.0029 0.0058   0.0127 0.0070
Locus3  0.3887 0.0647 0.0721   0.5049 0.4664
Locus4  0.0882 0.0290 0.0415   0.1429 0.1058
Locus5  0.2683 0.0266 0.0287   0.3414 0.3219
Locus6  0.1536 0.0306 0.0368   0.2143 0.1843
Locus7  0.0533 0.0202 0.0315   0.0935 0.0640
Locus8  0.0255 0.0192 0.0724   0.1008 0.0306
Locus9  0.3616 0.0244 0.0255   0.4484 0.4340
Locus10 0.3585 0.0522 0.0576   0.4630 0.4302
Global      NA     NA 0.0493   0.2163 0.1757
```

loci
A list of locus names
H_st
Between subpopulation heterozygsity per locus
D_st
Absolute differentiation per locus [11]
G_st
F_st analogue for multiple alleles per locus [11]
G_hed_st
Hedrick's standardized "differention" per locus [13]
D_jost
Jost's true allelic differentiation per locus [14]

Returned values cont.

`$estimate` A matrix containing identical data to the *Estimated_stats* worksheet in the `.xlsx` workbook or the *Estimated-stats[div.part].txt* text file. The last row in this matrix represents statistics calculate across all population samples and loci.

```
        Harmonic_N H_st_est D_st_est G_st_est G_hed_st_est D_Jost_est  Fis_WC
Locus1     43.1218   0.6841   0.0160   0.0234       0.0799     0.0578  0.0363
Locus2     43.5209   0.5035  -0.0019  -0.0038      -0.0084    -0.0046 -0.0474
Locus3     43.6403   0.8998   0.0566   0.0629       0.4688     0.4332  0.0266
Locus4     43.4476   0.7012   0.0225   0.0321       0.1134     0.0840  0.0205
Locus5     42.7674   0.9291   0.0177   0.0191       0.2542     0.2397  0.0539
Locus6     43.4476   0.8329   0.0228   0.0274       0.1675     0.1441  0.2010
Locus7     43.4476   0.6429   0.0142   0.0221       0.0670     0.0459  0.0173
Locus8     43.2566   0.2657   0.0168   0.0632       0.0884     0.0268  0.1976
Locus9     43.0673   0.9587   0.0153   0.0160       0.3352     0.3244  0.0407
Locus10    43.2469   0.9083   0.0439   0.0483       0.4181     0.3885  0.0448
Global          NA       NA       NA   0.0397       0.1806     0.1462  0.0655
         Fst_WC  Fit_WC
Locus1   0.0257  0.0610
Locus2  -0.0042 -0.0518
Locus3   0.0745  0.0991
Locus4   0.0357  0.0555
Locus5   0.0222  0.0749
Locus6   0.0300  0.2250
Locus7   0.0258  0.0427
Locus8   0.0689  0.2529
Locus9   0.0189  0.0588
Locus10  0.0564  0.0986
Global   0.0456  0.1081
```

loci

A list of locus names

Harmonic_N
Harmonic mean number of individuals typed per locus
H_st_est
Estimator of between subpopulation heterozygosity [12]
D_st_est
Estimator of absolute differentiation [12]
G_st_est
Nearly unbiased estimator of G_st [12]
G_hed_st_est
Estimator of Hedrick's G'_st [13]
D_Jost_est
Estimator of Jost's D [14]
Fis_WC
Weir and Cockerham's inbreeding coefficient estimator [16]
Fst_WC
Weir and Cockerham's fixation index estimator [16]
Fit_WC
Weir and Cockerham's overall fixation index estimator [16]

Returned values cont.

`$pairwise` A list of six (`WC_Fst = FALSE`) nine (`WC_Fst = TRUE`) matrices containing pairwise diversity statistics without bootstrapped confidence intervals.

```
[1] Gst
```

```
       pop1,  pop2, pop3, pop4,
pop1,     NA     NA    NA    NA
pop2, 0.0077     NA    NA    NA
pop3, 0.0401 0.0351    NA    NA
pop4, 0.0349 0.0307 0.009    NA
```

```
[1] G_hed_st
```

```
       pop1,  pop2,  pop3, pop4,
pop1,     NA     NA     NA    NA
pop2, 0.0486     NA     NA    NA
pop3, 0.2562 0.2293     NA    NA
pop4, 0.2271 0.2041 0.0606    NA
```

```
[1] D_Jost
```

```
       pop1,  pop2,  pop3, pop4,
pop1,     NA     NA     NA    NA
pop2, 0.0409     NA     NA    NA
pop3, 0.2254 0.2011     NA    NA
pop4, 0.1989 0.1790 0.0519    NA
```

```
[1] Gst_est
```

```
       pop1,  pop2,  pop3, pop4,
pop1,     NA     NA     NA    NA
pop2, 0.0019     NA     NA    NA
pop3, 0.0341 0.0287     NA    NA
pop4, 0.0296 0.0251 0.0032    NA


[1] G_hed_st_est


       pop1,  pop2,  pop3, pop4,
pop1,     NA     NA     NA    NA
pop2, 0.0124     NA     NA    NA
pop3, 0.2264 0.1954     NA    NA
pop4, 0.1992 0.1732 0.0224    NA


[1] D_Jost_est


       pop1,  pop2,  pop3, pop4,
pop1,     NA     NA     NA    NA
pop2, 0.0027     NA     NA    NA
pop3, 0.1803 0.1579     NA    NA
pop4, 0.1484 0.1325 0.0102    NA


[1] Fis_WC


       pop1,  pop2,  pop3, pop4,
pop1,     NA     NA     NA    NA
pop2, 0.0908     NA     NA    NA
pop3, 0.0723 0.0832     NA    NA
pop4, 0.0711 0.0806 0.0635    NA


[1] Fst_WC


       pop1,  pop2,  pop3, pop4,
pop1,     NA     NA     NA    NA
```

```
pop2, 0.0027     NA     NA    NA
pop3, 0.0647 0.0543     NA    NA
pop4, 0.0563 0.0478 0.0057    NA
```

```
[1] Fit_WC
```

```
       pop1,  pop2,  pop3, pop4,
pop1,     NA     NA     NA    NA
pop2, 0.0933     NA     NA    NA
pop3, 0.1323 0.1331     NA    NA
pop4, 0.1233 0.1245 0.0689    NA
```

Returned values cont.

`$bs_locus` A list containing six (`WC_Fst = FALSE`) - nine (`WC_Fst = TRUE`) matrices of locus values for each estimated statistic, along with their respective 95% confidence interval.

```
[1] Gst
```

```
       Actual Lower_CI Upper_CI
Locus1 0.0328   0.0090   0.0566
Locus2 0.0058  -0.0044   0.0160
Locus3 0.0721   0.0577   0.0865
global 0.0493   0.0456   0.0530
```

```
[1] G_hed_st
```

```
       Actual Lower_CI Upper_CI
Locus1 0.1093   0.0405   0.1781
Locus2 0.0127  -0.0090   0.0344
Locus3 0.5049   0.4482   0.5616
global 0.2163   0.2026   0.2300
```

```
[1] D_Jost
```

```
       Actual Lower_CI Upper_CI
Locus1 0.0791   0.0294   0.1288
Locus2 0.0070  -0.0048   0.0188
Locus3 0.4664   0.4127   0.5201
global 0.1757   0.1642   0.1872
```

```
[1] Gst_est
```

```
        Actual Lower_CI Upper_CI
Locus1  0.0234  -0.0005   0.0473
Locus2 -0.0038  -0.0141   0.0065
Locus3  0.0629   0.0484   0.0774
global  0.0397   0.0360   0.0434
```

```
[1] G_hed_st_est
```

```
        Actual Lower_CI Upper_CI
Locus1  0.0799   0.0076   0.1522
Locus2 -0.0084  -0.0309   0.0141
Locus3  0.4688   0.4057   0.5319
global  0.1806   0.1662   0.1950
```

```
[1] D_Jost_est
```

```
        Actual Lower_CI Upper_CI
Locus1  0.0578   0.0056   0.1100
Locus2 -0.0046  -0.0169   0.0077
Locus3  0.4332   0.3737   0.4927
global  0.1462   0.1313   0.1611
```

```
[1] Fis_WC
```

```
        Actual Lower_CI Upper_CI
Locus1  0.0363  -0.0281   0.1007
Locus2 -0.0474  -0.1487   0.0539
Locus3  0.0266  -0.0089   0.0621
global  0.0655   0.0504   0.0806
```

```
[1] Fst_WC
```

```
        Actual Lower_CI Upper_CI
Locus1  0.0257  -0.0008   0.0522
```

```
Locus2 -0.0042  -0.0176   0.0092
Locus3  0.0745   0.0576   0.0914
global  0.0456   0.0412   0.0500
```

```
[1] Fit_WC
```

```
        Actual Lower_CI Upper_CI
Locus1  0.0610  -0.0069   0.1289
Locus2 -0.0518  -0.1490   0.0454
Locus3  0.0991   0.0569   0.1413
global  0.1081   0.0965   0.1197
```

Returned values cont.

`$bs_pairwise` A list containing six (`WC_Fst = FALSE`) - nine (`WC_Fst = TRUE`) matrices of pairwise values for each estimated statistic, along with their respective 95% confidence interval.

```
[1] Gst
```

```
                Actual Lower_CI Upper_CI
pop1, vs. pop2, 0.0077   0.0055   0.0099
pop1, vs. pop3, 0.0401   0.0336   0.0466
pop1, vs. pop4, 0.0349   0.0301   0.0397
pop5, vs. pop6, 0.0281   0.0239   0.0323
```

```
[1] G_hed_st
```

```
                Actual Lower_CI Upper_CI
pop1, vs. pop2, 0.0486   0.0357   0.0615
pop1, vs. pop3, 0.2562   0.2239   0.2885
pop1, vs. pop4, 0.2271   0.2016   0.2526
pop5, vs. pop6, 0.1943   0.1660   0.2226
```

```
[1] D_Jost
```

```
                Actual Lower_CI Upper_CI
pop1, vs. pop2, 0.0409   0.0299   0.0519
pop1, vs. pop3, 0.2254   0.1964   0.2544
pop1, vs. pop4, 0.1989   0.1762   0.2216
pop5, vs. pop6, 0.1710   0.1451   0.1969
```

```
[1] Gst_est
```

```
                Actual Lower_CI Upper_CI
pop1, vs. pop2, 0.0019  -0.0003   0.0041
pop1, vs. pop3, 0.0341   0.0275   0.0407
pop1, vs. pop4, 0.0296   0.0249   0.0343
pop5, vs. pop6, 0.0217   0.0175   0.0259
```

```
[1] G_hed_st_est
```

```
                Actual Lower_CI Upper_CI
pop1, vs. pop2, 0.0124  -0.0010   0.0258
pop1, vs. pop3, 0.2264   0.1920   0.2608
pop1, vs. pop4, 0.1992   0.1732   0.2252
pop5, vs. pop6, 0.1568   0.1279   0.1857
```

```
[1] D_Jost_est
```

```
                Actual Lower_CI Upper_CI
pop1, vs. pop2, 0.0027  -0.0100   0.0154
pop1, vs. pop3, 0.1803   0.1493   0.2113
pop1, vs. pop4, 0.1484   0.1176   0.1792
pop5, vs. pop6, 0.1199   0.1006   0.1392
```

```
[1] Fis_WC
```

```
                Actual Lower_CI Upper_CI
pop1, vs. pop2, 0.0908   0.0629   0.1187
pop1, vs. pop3, 0.0723   0.0488   0.0958
pop1, vs. pop4, 0.0711   0.0581   0.0841
pop5, vs. pop6, 0.0420   0.0274   0.0566
```

```
[1] Fst_WC
```

```
                Actual Lower_CI Upper_CI
pop1, vs. pop2, 0.0027  -0.0017   0.0071
```

```
pop1, vs. pop3, 0.0647   0.0525   0.0769
pop1, vs. pop4, 0.0563   0.0476   0.0650
pop5, vs. pop6, 0.0417   0.0338   0.0496
```

```
[1] Fit_WC
```

```
                Actual Lower_CI Upper_CI
pop1, vs. pop2, 0.0933   0.0671   0.1195
pop1, vs. pop3, 0.1323   0.1115   0.1531
pop1, vs. pop4, 0.1233   0.1076   0.1390
pop5, vs. pop6, 0.0820   0.0694   0.0946
```

4.2 in.calc()

The general usage of this function is as follows:

```
in.calc(infile, outfile = NULL, gp = 3, bs_locus = FALSE,
        bs_pairwise = FALSE, bootstraps = 0, Plot = FALSE
        parallel = FALSE)
```

4.2.1 Arguments

`infile` Specifying the name of the *‘genepop’* [22] file from which the statistics are to be calculated This file can be in either the 3 digit of 2 digit format. The name must be a character string.

`outfile` Allows useRs to specify a suffix for output folder and files. Name must a character string enclosed in either “” or ‘’.

`gp` Specifies the digit format of the `infile`. Either 3 (default) or 2.

`bs_locus` Gives useRs the option to bootstrap locus statistics. Results will be written to *.xlsx* file by default if the package `xlsx` is installed, and to a *.png* file if `Plot=TRUE`. If `xlsx` is not installed, results will be written to *.txt* files.

`bs_pairwise` Gives useRs the option to bootstrap statistics across all loci for each pairwise population comparison. Results will be written to a *.xlsx* file by default if the package `xlsx` is installed. If `xlsx` is not installed, results will be written to *.txt* files.

Arguments cont.

`bootstraps` Determines the number of bootstrap iterations to be carried out. The default value is `bootstraps = 0`, this is only valid when all bootstrap options are false. There is no upper limit on the number of bootstrap iterations, however very large numbers of bootstrap iterations for pairwise calculations ($>$ 1000) may take a long time to run for large data sets.

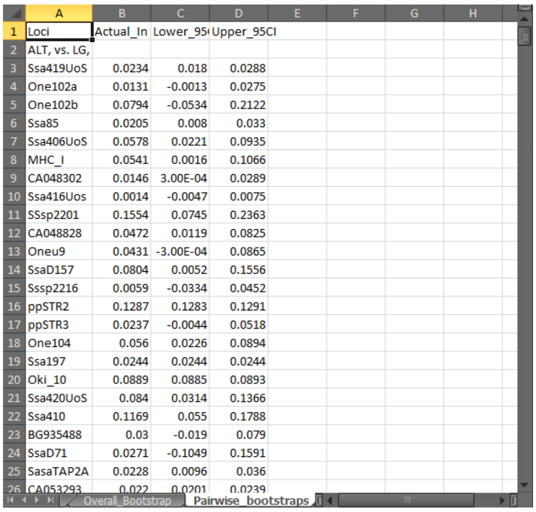
`Plot` Optional `.png` image file of the plotted bootstrap results for locus I_n if `bs_locus = TRUE`. The default option is `Plot = FALSE`.

`parallel` A logical argument specifying if computations should be run in parallel on all available CPU cores. If `parallel = TRUE`, batches of jobs will be distributed to all cores resulting in faster completion. In Windows, the packages `doParallel`, `iterators`, `parallel` and `foreach` must be installed to use parallel computation. In Linux the packages `doSNOW`, `parallel`, `snow`, `iterators` and `foreach` should be installed.

4.2.2 Returned values

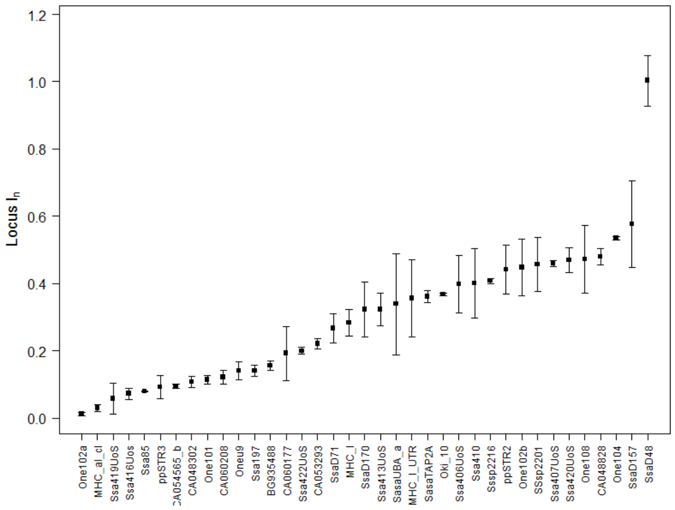
Values returned from `in.calc` are a single `.xlsx` workbook (if the package `xlsx` is installed), containing between one to three worksheets, (`In_allele_stats` by default or separate *.txt* files (if `xlsx` is unavailable). If `Plot = TRUE` an additional *.png* plot file will be written. An example of a *.xlsx* workbook and a *.png* plot are given below:

Example of bootstrapped locus I_n results



Loci	Actual_In	Lower_95(	Upper_95CI
ALT, vs. LG,			
Ssa419UoS	0.0234	0.018	0.0288
One102a	0.0131	-0.0013	0.0275
One102b	0.0794	-0.0534	0.2122
Ssa85	0.0205	0.008	0.033
Ssa406UoS	0.0578	0.0221	0.0935
MHC_I	0.0541	0.0016	0.1066
CA048302	0.0146	3.00E-04	0.0289
Ssa416Uos	0.0014	-0.0047	0.0075
SSsp2201	0.1554	0.0745	0.2363
CA048828	0.0472	0.0119	0.0825
Oneu9	0.0431	-3.00E-04	0.0865
SsaD157	0.0804	0.0052	0.1556
Sssp2216	0.0059	-0.0334	0.0452
ppSTR2	0.1287	0.1283	0.1291
ppSTR3	0.0237	-0.0044	0.0518
One104	0.056	0.0226	0.0894
Ssa197	0.0244	0.0244	0.0244
Oki_10	0.0889	0.0885	0.0893
Ssa420UoS	0.084	0.0314	0.1366
Ssa410	0.1169	0.055	0.1788
BG935488	0.03	-0.019	0.079
SsaD71	0.0271	-0.1049	0.1591
SasaTAP2A	0.0228	0.0096	0.036
CA053293	0.022	0.0201	0.0239

Example of bootstrapped locus I_n results plot



Returned values cont.

For useRs wishing to carry out post analysis manipulations, all results from `in.calc` are returned to the `R` environment. Depending on the bootstrap options chosen these results include between one to three of the `variables` below:

`Allele_In` A character matrix of allelic I_n values per locus along with locus sums.

```
        Allele.1 Allele.2 Allele.3 Allele.4 Allele.5 Sum
Locus1  0.0036   0.0036   0.0144   0.004    0.0178   0.0581
Locus2  0.0095   0.0015   0.0013                     0.0123
Locus3  0.0473   0.004    0.0098   0.0234   0.027    0.4482
Locus4  0.0032   0.0029   0.0053   0.0135   0.0109   0.08
Locus5  0.0111   0.0029   0.0042   0.0045   0.0044   0.3983
Locus6  0.0394   0.0379   0.0181   0.005    0.0352   0.2839
Locus7  0.0077   0.0131   0.0046   0.0087   0.0166   0.1068
Locus8  0.0157   0.0469   0.0054   0.0048            0.0728
Locus9  0.0107   0.0075   0.0069   0.0054   0.0081   0.4571
Locus10 0.0038   0.0232   0.0091   0.0326   0.0295   0.4799
```

Each row of this results matrix represents each locus in the `infile`. Each column represents the allele specific I_n per locus except the last column, which contains the sum of allele I_n for each locus.

Returned values cont.

`l_bootstrap` A character matrix of locus *In* values as well as 95% confidence intervals, calculated from bootstraps (Manly, 1997). Returned when `bs_locus = TRUE`.

```
            In Lower_95CI Upper_95CI
Locus1  0.0581     0.0180     0.0982
Locus2  0.0123    -0.0016     0.0262
Locus3  0.4482     0.3964     0.5000
Locus4  0.0800     0.0525     0.1075
Locus5  0.3983     0.3178     0.4788
Locus6  0.2839     0.2450     0.3228
Locus7  0.1068     0.0843     0.1293
Locus8  0.0728     0.0233     0.1223
Locus9  0.4571     0.3928     0.5214
Locus10 0.4799     0.4381     0.5217
```

Each row in this matrix represents each locus. The first column is the locus sum I_n as in the final column in `Allele_In`. The second and third columns represent the lower and upper confidence intervals per locus respectively.

`PW_bootstrap` A list of matrices for each pairwise population comparison of bootstrapped pairwise locus I_n values.

```
[1] pop1, vs. pop2,
```

```
           In Lower_95CI Upper_95CI
Locus1 0.0234    -0.0021     0.0489
Locus2 0.0131    -0.0074     0.0336
Locus3 0.0794     0.0163     0.1425
Locus4 0.0205    -0.0035     0.0445
Locus5 0.0578     0.0091     0.1065
```

```
[1] pop1, vs. pop3,
```

```
           In Lower_95CI Upper_95CI
Locus1 0.0167     0.0058     0.0276
Locus2 0.0115    -0.0028     0.0258
Locus3 0.3157     0.2167     0.4147
Locus4 0.0982     0.0433     0.1531
Locus5 0.2427     0.1717     0.3137
```

```
[1] pop1, vs. pop4,
```

```
           In Lower_95CI Upper_95CI
Locus1 0.0233    -0.0069     0.0535
Locus2 0.0112     0.0016     0.0208
Locus3 0.3395     0.2556     0.4234
Locus4 0.0419    -0.0074     0.0912
Locus5 0.2794     0.2205     0.3383
```

```
[1] pop1, vs. pop5,
           In Lower_95CI Upper_95CI
Locus1 0.0619     0.0330     0.0908
Locus2 0.0118    -0.0015     0.0251
Locus3 0.3690     0.2694     0.4686
Locus4 0.0630    -0.0223     0.1483
Locus5 0.2615     0.1582     0.3648
```

```
[1] pop1, vs. pop6,
```

```
           In Lower_95CI Upper_95CI
Locus1 0.0264    -0.0053     0.0581
Locus2 0.0123    -0.0026     0.0272
Locus3 0.2815     0.2145     0.3485
Locus4 0.0297     0.0022     0.0572
Locus5 0.2187     0.1358     0.3016
```

4.3 `readGenepop.user()`

The general usage of `readGenepop.user` is:

```
readGenepop.user(infile = NULL, gp = 3, bootstrap = FALSE)
```

4.3.1 Arguments

`infile` Specifying the name of the *'genepop'* [22] file from which the statistics are to be calculated This file can be in either the 3 digit of 2 digit format. The name must be a character string.

`gp` Specifies the digit format of the `infile`. Either 3 (default) or 2.

`bootstrap` A logical argument indicating whether the `infile` should be sampled with replacement and all returned parameters calculated from this bootstrapped data.

4.3.2 Returned values

`npops` The number of population samples in `infile`.

`nloci` The number of loci in `infile`.

`pop_alleles` A list of two matrices per population. Each matrix per population contains haploid allele designations.

`pop_list` A list of matrices (n = `npops`) containing the diploid genotypes of individuals per locus.

`loci_names` A character vector containing the names of loci from `infile`.

`pop_pos` A numeric vector or the row index locations of the first individual per population in `infile`.

`pop_sizes` A numeric vector of length `npops` containing the number of individuals per population sample in `infile`.

`allele_names` A list of `npops` lists containing `nloci` character vectors of alleles names per locus. Useful for identifying unique alleles.

`all_alleles` A list of `nloci` character vectors of all alleles observed across all population samples in `infile`.

`allele_freq` A list containing `nloci` matrices containing allele frequencies per alleles per population sample.

`raw_data` An unaltered data frame of `infile`.

`loci_harm_N` A numeric vector of length `nloci`, containing the harmonic mean number of individuals genotyped per locus.

`n_harmonic` A numeric value representing the harmonic mean of `npops`.

`pop_names` A character vector containing a six character population sample name for each population in `infile` (the first six characters of the first individual).

`indtyp` A list of length `nloci` containing character vectors of length `npops`, indicating the number of individuals per population sample typed at each locus.

`nalleles` A vector of the total number of alleles observed at each locus.

`bs_file` A dataframe/genpop object of bootstrapped data. Returned if `bootstrap = TRUE`.

`obs_all...` A list of matrices of the observed number of allele occurrences per population.

4.4 corPlot()

The general usage of corPlot is:

```
corPlot(x,y)
```

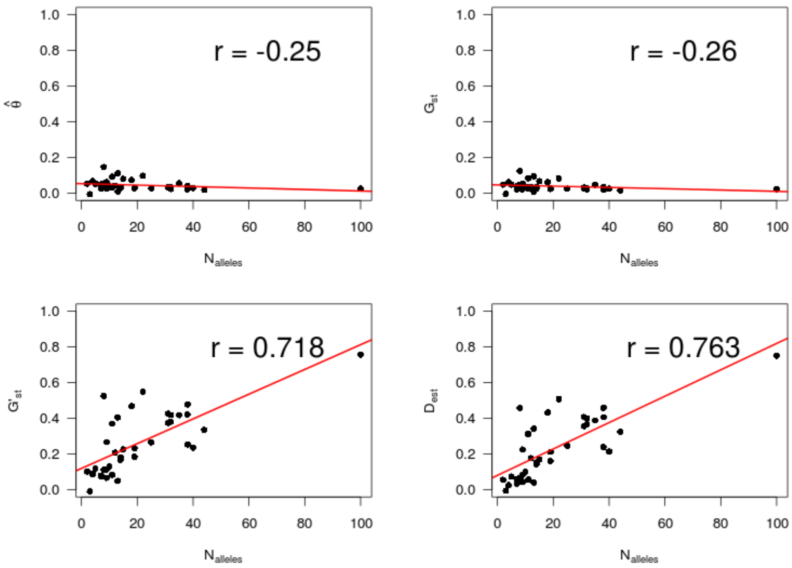
4.4.1 Arguments

x	The object returned by the function readGenepop.user.
y	The object returned by the function div.part.

4.4.2 Returned values

plot	A console plot is automatically created using this functions. As the plot is intended for exploratory purposes, it is not written to file. UseRs can save the lot manually if required. below is an example of the returned plot.

Returned plot from the function `corPlot`



The plot depicts the relationship between the estimated statistics calculated by `div.part` and the number of alleles per locus. Lines represents the line of best fit. Pearson's product-moment correlation coefficient is also provided.

4.5 `difPlot()`

The general usage of `difPlot` is:

```
difPlot(x,y)
```

4.5.1 Arguments

`x` The object returned by the function `div.part`.

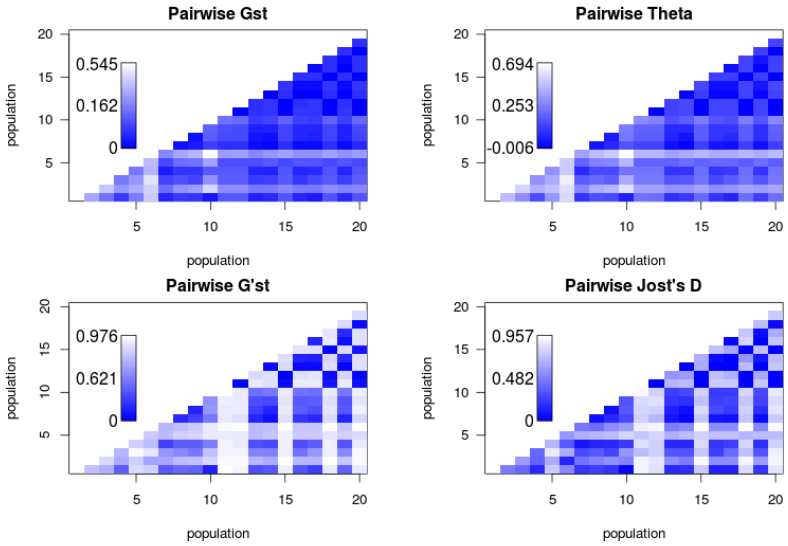
`outfile` A folder name or directory indicating where interactive plots should be written. It is advisable, though not essential that this argument be set to the same `outfile` argument as for `div.part`. This argument is only valid when `interactive = TRUE`. If no argument is given for `outfile`, while `interactive = TRUE`, plot files will be written to the working directory. Folder name should be given as a character string.

`interactive` A logical argument indicating whether useRs would like to plot their results to interactive *.html* files produced by `sendplot`. `TRUE` indicates that results should be written to file, whereas `FALSE` indicates that results should be plotted to the R graphics device.

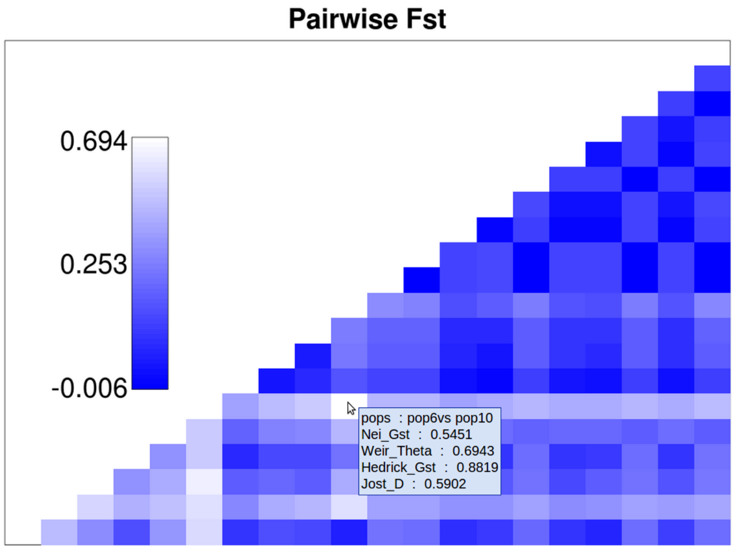
4.5.2 Returned values

`Plot` Depending on the argument given for `interactive`, either a single plot will be passed to the `R` graphic device (i.e. when `interactive = FALSE`) or 3-4 *.html* files will be written to a user defined location.

Returned plot from the function `difPlot` when `interactive =FALSE`



One of the returned plots from the function `difPlot` when `interactive =TRUE`



pops : pop6vs pop10
Nei_Gst : 0.5451
Weir_Theta : 0.6943
Hedrick_Gst : 0.8819
Jost_D : 0.5902

As can be seen, the plots produced when `interactive = TRUE` are much more useful than when `interactive = FALSE`, due to useRs ability to identify population comparisons of interest. These plots contain tool-tip information, courtesy of the `sendplot` package.

5 Examples

In this section worked examples of each of the three functions documented above are given. The examples will employ the test data set distributed with `diveRsity`, `Test_data`. Care has been take to ensure that examples can be used independently, thus some processes are repeated for each function examples, such as loading `Test_data` into the `R` session.
N.B. All examples assume that you have already downloaded, installed and loaded `diveRsity`.

5.1 div.part

This example is specific to the function `div.part`. It has been written to demonstrate way in the which the function may be used. It has not been written as an exhaustive demonstration.

5.1.1 Setting your working directory

In any `R` session it is sensible to have a folder on your system where any output files etc. are to be written. When using `diveRsity`, it is recommended that you set your `working directory` to the location of your input file.
To set your working directory, use:

```
setwd("mypath")
```

Simply replace 'mypath' with your actual file path. Make sure to use '/' or '\\' to separate directory levels (e.g. c:/Users/Kevin/etc., or c:\\Users \\Kevin \\etc.). `R` does not recognise the '\' symbol for pathways.

5.1.2 Loading Test_data

Test_data is only required for these examples. UseRs should replace the argument 'infile = Test_data' with 'infile = "myfilename"' when wishing to analyse their own data set.

```
> data(Test_data, package = "diveRsity")
```

This command loads Test_data into the current R session.

5.1.3 Running div.part

To run div.part, where locus bootstrap and pairwise bootstrap results are returned without plotting, use the following:

```
> div_results <- div.part(infile = Test_data, outfile = "Test",
+                          gp = 3, WC_Fst = TRUE, bs_locus = TRUE,
+                          bs_pairwise = TRUE, bootstraps = 3,
+                          Plot = FALSE, parallel = TRUE)
```

```
[NOTE]
Cores successfully registered for parallel computations...
```

N.B. in this example bootstraps = 100 to reduce the time taken to run the example.
When the analysis has finished a folder named Test-[diveRsity] should be written to your working directory. This folder will contain either a single *.xlsx* workbook named '*[div.part].xlsx*' (if xlsx is installed), or four *.txt* files named, '*Standard-stats[div.part].txt*', '*Estimated-stats[div.part].txt*', '*Locus-bootstrap[div.part].txt*' and '*Pairwise-bootstrap[div.part].txt*' if it is not.

5.1.4 Accessing your results within the R session

All of the results written to file are also assigned to the variable test_results. To access these results it is useful to understand the structure of the objects test_results contains. Although the objects have been described in the **Returned values** section for div.part, a further visual description will be provided here.
Using the following will show you the names of all objects within test_results:

```
> names(div_results)
```

```
[1] "standard"    "estimate"    "pairwise"    "bs_locus"
[5] "bs_pairwise"
```

To access an object within `test_results` you can use the extract operator '`$`'. For example, if you want to know what type of object `bs_locus` is, use:

```
> typeof(div_results$bs_locus)
```

```
[1] "list"
```

From the **Returned values** section for `div.part`, it is known that `bs_locus` is indeed a list containing six matrices. This object can be explored further using:

```
> names(div_results$bs_locus)
```

```
[1] "Gst"          "G_hed_st"     "D_Jost"
[4] "Gst_est"      "G_hed_st_est" "D_Jost_est"
[7] "Fis_WC"       "Fst_WC"       "Fit_WC"
```

Each of the named objects within `test_results$bs_locus` are known to be matrices from above. This means that we can use matrix indexing to access any of the information within any of the matrices. In `R`, to access a specific value within a matrix, we only need to know the row and column that the value is in. If we wanted to access a value that lies in the 5^{th} row and the 1^{st} column the following command could be used:

```
mymatrix[5, 1]
```

The first digit within the '[ ]' (i.e. before the ',') in R always refers to the **row** location of a value and the second to the **column** location.
It is possible to access more than one value in a matrix using indexing. If we wanted to look at the first 10 rows of test_resultsbs_locusGst, we would use the following code.

```
> div_results$bs_locus$Gst[1:10, ]
```

```
        Actual Lower_CI Upper_CI
Locus1  0.0328   0.0175   0.0481
Locus2  0.0058  -0.0135   0.0251
Locus3  0.0721   0.0634   0.0808
Locus4  0.0415   0.0275   0.0555
Locus5  0.0287   0.0172   0.0402
Locus6  0.0368   0.0295   0.0441
Locus7  0.0315   0.0056   0.0574
Locus8  0.0724   0.0360   0.1088
Locus9  0.0255   0.0197   0.0313
Locus10 0.0576   0.0383   0.0769
```

By leaving the column index blank (i.e. no numbers after the ','), all columns are returned. Similarly, if we wanted to view all values in the first column of test_resultsbs_locusGst, we would use:

```
div_results$bs_locus$Gst[ ,1]
```

The other values returned by div.part can be accessed in a similar fashion. When you understand how to access the results within R, many *post-analysis* processes can be used such as correlations, regressions and plotting.

5.2 in.calc

This example is specific to the function in.calc. It has been written to demonstrate way in the which the function may be used. It has not been written as an exhaustive demonstration.

5.2.1 Setting your working directory

In any R session it is sensible to have a folder on your system where any output files etc. are to be written. When using diveRsity, it is recommended that you set your working directory to the location of your input file.
To set your working directory, use:

```
setwd("mypath")
```

Simply replace 'mypath' with your actual file path. Make sure to use '/' or '\\' to separate directory levels (e.g. c:/Users/Kevin/etc., or c:\\Users\\Kevin \\etc.). R does not recognise the '\' symbol for pathways.

5.2.2 Loading Test_data

Test_data is only required for these examples. UseRs should replace the argument 'infile = Test_data' with 'infile = "myfilename"' when wishing to analyse their own data set.

```
> data(Test_data, package = "diveRsity")
```

This command loads `Test_data` into the current `R` session.

5.2.3 Running `in.calc`

To run `in.calc`, where locus bootstrap and pairwise bootstrap results are returned without plotting, use the following:

```
> in_results <- in.calc (infile = Test_data, outfile = "Test",
+                           gp = 3, bs_locus = TRUE,
+                           bs_pairwise = TRUE, bootstraps = 3,
+                           Plot = FALSE, parallel = TRUE)
```

N.B. in this example `bootstraps = 100` to reduce the time taken to run the example.
When the analysis has finished a folder named `Test-[diveRsity]` should be written to your working directory. This folder will contain either a single *.xlsx* workbook named '*[].xlsx*' (if `xlsx` is installed), or three *.txt* files named, '*Allele-In[in.calc].txt*', '*Overall-bootstrap[in.calc].txt*' and '*Pairwise-bootstrap[in.calc].txt*' if it is not.

5.2.4 Accessing your results within the `R` session

All of the results written to file are also assigned to the variable `test_results`. To access these results it is useful to understand the structure of the objects `test_results` contains. Although the objects have been described in the **Returned values** section for `in.calc`, a further visual description will be provided here.

Using the following will show you the names of all objects within `test_results`:

```
> names(in_results)
```

```
[1] "Allele_In"    "l_bootstrap"  "PW_bootstrap"
```

To access an object within `test_results` you can use the extract operator '`$`'. For example, if you want to know what type of object `PW_bootstrap` is, use:

```
> typeof(in_results$PW_bootstrap)
```

```
[1] "list"
```

From the **Returned values** section for `in.calc`, it is known that `PW_bootstrap` is indeed a list of matrices of bootstrapped locus results for each pairwise comparison. To find the names of the matrices within `PW_bootstraps`, use:

```
> names(in_results$PW_bootstrap)
```

```
 [1] "pop1, vs. pop2," "pop1, vs. pop3," "pop1, vs. pop4,"
 [4] "pop1, vs. pop5," "pop1, vs. pop6," "pop2, vs. pop3,"
 [7] "pop2, vs. pop4," "pop2, vs. pop5," "pop2, vs. pop6,"
[10] "pop3, vs. pop4," "pop3, vs. pop5," "pop3, vs. pop6,"
[13] "pop4, vs. pop5," "pop4, vs. pop6," "pop5, vs. pop6,"
```

From this we see that `PW_bootstrap` contains 15 matrices for each of the 15 possible pairwise comparisons from the six population samples in `Test_data`. We can explore any of these matrices using matrix indexing. In `R`, to access a specific value within a matrix, we only need to know the row and column that the value is in (i.e. its index). If we wanted to access a value that lies in the 5th row and the 1st column the following command could be used:

```
mymatrix[5, 1]
```

The first digit within the '[ ]' (i.e. before the ',') in `R` always refers to the **row** location of a value and the second to the **column** location.
To look at the first 3 rows of the comparison between pop1 and pop2 in `PW_bootstrap`, we would use the following code.

```
> in_results$PW_bootstrap[["pop1, vs. pop2,"]][1:3, ]
```

```
           In Lower_95CI Upper_95CI
Locus1 0.0234     0.0040     0.0428
Locus2 0.0131     0.0063     0.0199
Locus3 0.0794     0.0590     0.0998
```

By leaving the column index blank (i.e. no numbers after the ','), all columns are returned. Similarly, if we wanted to view all values in the first column of `test_results$PW_bootstrap[["pop1, vs.  pop2,"]]`, we would use:

```
in_results$PW_bootstrap[["pop1, vs. pop2,"]][ ,1]
```

The other values returned by `in.calc` can be accessed in a similar fashion. When you understand how to access the results within `R`, many *post-analysis* processes can be used such as correlations, regressions and plotting.

5.3 `readGenepop.user`

This example is specific to the function `readGenepop.user`. It has been written to demonstrate way in the which the function may be used. It has not been written as an exhaustive demonstration.

5.3.1 Setting your `working directory`

In any `R` session it is sensible to have a folder on your system where any output files etc. are to be written. When using `diveRsity`, it is recommended that you set your `working directory` to the location of your input file.
To set your working directory, use:

```
setwd("mypath")
```

Simply replace ‘mypath’ with your actual file path. Make sure to use ‘/’ or ‘\\’ to separate directory levels (e.g. c:/Users/Kevin/etc., or c:\\Users\\Kevin \\etc.). `R` does not recognise the ‘\’ symbol for pathways.

5.3.2 Loading `Test_data`

`Test_data` is only required for these examples. UseRs should replace the argument ‘`infile = Test_data`’ with ‘`infile = "myfilename"`’ when wishing to analyse their own data set.

```
> data(Test_data, package = "diveRsity")
```

This command loads Test_data into the current R session.

5.3.3 Running readGenepop.user

To run readGenepop.user without producing a bootstrap file, use:

```
> gp_res <- readGenepop.user(infile = Test_data, gp = 3,
+                            bootstrap = FALSE)
```

5.3.4 Accessing your results within the R session

The readGenepop.user function does not write anything to file. Instead results are only returned to the R environment.
To explore what these results are, use:

```
> names(gp_res)
```

```
 [1] "npops"          "nloci"          "pop_alleles"
 [4] "pop_list"       "loci_names"     "pop_pos"
 [7] "pop_sizes"      "allele_names"   "all_alleles"
[10] "allele_freq"    "raw_data"       "loci_harm_N"
[13] "n_harmonic"     "pop_names"      "indtyp"
[16] "nalleles"       "obs_allele_num"
```

For a description of each of these objects see section **4.3.2**.

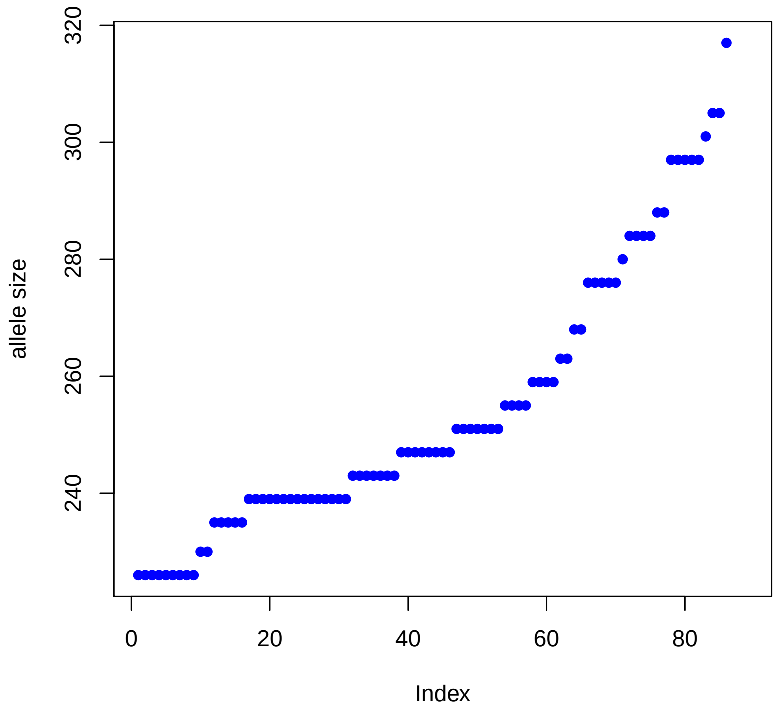
5.3.5 Applications for `readGenepop.user`

`readGenepop.user` is not like the other two function in that the results returned have no particularly informative format. Instead the results are the building blocks to developing other analysis methods for useRs who may not have the necessary programming skills to extract such information from genetic data. In this section two examples of applications of `readGenepop.user` are provided. UseRs are encouraged to use the function to develop their own methods.

'*Ad hoc*' investigation of locus mutation model

Understanding the likely mutation model a particular microsatellite locus follows is important for a range of analyses which make explicit assumptions. One way to ensure your data does not violate these assumption is to visualise the allele distribution at loci and assess whether the pattern fits the expectation of a given model.
`readGenepop.user` returns an object `pop_alleles` which contains npops x 2 matrices. Each matrix contains a haploid genotype per individual per locus, and every two matrices correspond to a single population sample. For example matrices 1 and 2 correspond to population sample 1, matrices 3 and 4 correspond to population sample 2 and so on. Using this object, it is possible to plot the allele size distribution to assess it allele fragments fit say the single step mutation model (SSM).

```
> locus18_pop1 <- c(gp_res$pop_alleles[[1]][[1]][,18],
+                   gp_res$pop_alleles[[2]][[1]][,18])
> # sort alleles by size
> allele_sort <- order(locus18_pop1, decreasing = FALSE)
> #plot
> plot(locus18_pop1[allele_sort], ylab = "allele size", col="blue",
+      pch = 16)
```



From this figure we could conclude that locus 18 in population 1 is likely to follow SSM given that allele size increases in a generally regular fashion. Any gaps are also a multiple of the repeat motif length.

Although this example is basic and does not have a rigorous statistical basis, the value of such data exploration is clear. Indeed, useRs with suitable knowhow could likely easily develop statistically valid model tests for this particular example.

5.3.6 A hypothetical example

This example is for illustrative purposes.

Say for some reason, we were interested in assessing the sampling properties of the number of alleles at a particular locus, `readGenepop.user` is ideal to do this. We will use Test_data for this example and the number of bootstrap iterations will be 1000. We know that Test_data contains 37 loci so we will have to be able to count the number of alleles for each of these in each bootstrap iteration.

The code

```
> # Define a results matrix with 37 columns (loci) and
> # 1000 rows (bootstraps)to record allele number per locus
>
> num_all <- matrix(rep(0,(37*10)), ncol = 37)
> # Now using readGenepop.user we can fill the matrix
> bs<-10
> for(i in 1:bs){
+     # first produce a bootstrap file
+
+     x <- readGenepop.user(infile = Test_data, gp = 3,
+                           bootstrap = TRUE)
+
+     # Now record the number of alleles at each locus
+
+     num_all[i, ] <- x$nalleles
+ }
> # Now we can use this data to calculate the mean
> # number of alleles per locus as well at their
> # 95% confidence intervals
>
> mean_num <- colMeans(num_all)
> lower<-vector()
> upper<-vector()
> for(i in 1:ncol(num_all)){
+     lower[i] <- mean_num[i] - (1.96 * sd(num_all[,i]))
+     upper[i] <- mean_num[i] + (1.96 * sd(num_all[,i]))
+ }
> # Now we can create a data frame of these results
>
> bs_res <- data.frame(mean_num, lower, upper)
> bs_res[1:10,]
```

```
   mean_num     lower     upper
1       6.0  4.399667  7.600333
2       2.9  2.280194  3.519806
3      17.9 17.280194 18.519806
4       7.6  6.229556  8.970444
5      35.2 32.459113 37.940887
6      13.9 13.280194 14.519806
7       8.5  7.114071  9.885929
8       4.0  4.000000  4.000000
9      41.0 38.736787 43.263213
10     33.4 31.293063 35.506937
```

This is perhaps not the most efficient way to do this kind of analysis but it does make it more accessible to non-programmers.

References

[1] R Development Core Team, "R: A Language and Environment for Statistical Computing," 2010.

[2] L. J, "Plotrix: a package in the red light district of r," *R-News*, vol. 6, no. 4, pp. 8–12, 2006.

[3] A. A. Dragulescu, *xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files*, 2012. R package version 0.4.2.

[4] L. A. Shepherd, D. P. Gaile, L. Sucheston, A. Bruno, and K. F. Manly, *sendplot: Tool for sending interactive plots with tool-tip content.*, 2012. R package version 3.8.10.

[5] R. Analytics, *doSNOW: Foreach parallel adaptor for the snow package*, 2012. R package version 1.0.6.

[6] R. Analytics, *doParallel: Foreach parallel adaptor for the parallel package*, 2012. R package version 1.0.1.

[7] L. Tierney, A. J. Rossini, N. Li, and H. Sevcikova, *snow: Simple Network of Workstations*, 2012. R package version 0.3-10.

[8] R Development Core Team, *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[9] R. Analytics, *foreach: Foreach looping construct for R*, 2012. R package version 1.4.0.

[10] R. Analytics, *iterators: Iterator construct for R*, 2012. R package version 1.0.6.

[11] M. Nei, "Analysis of gene diversity in subdivided populations," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 70, no. 12, p. 3321, 1973.

[12] M. Nei and R. Chesser, "Estimation of fixation indices and gene diversities," *Ann. Hum. Genet*, vol. 47, no. Pt 3, pp. 253–259, 1983.

[13] P. Hedrick, "A standardized genetic differentiation measure," *Evolution*, vol. 59, no. 8, pp. 1633–1638, 2005.

[14] L. Jost, “G ST and its relatives do not measure differentiation,” *Molecular Ecology*, vol. 17, no. 18, pp. 4015–4026, 2008.

[15] A. Chao, L. Jost, S. Chiang, Y. Jiang, and R. Chazdon, “A two-stage probabilistic approach to multiple-community similarity indices.,” *Biometrics*, vol. 64, no. 4, pp. 1178–86, 2008.

[16] B. Weir and C. Cockerham, “Estimating F-statistics for the analysis of population structure,” *Evolution*, vol. 38, no. 6, pp. 1358–1370, 1984.

[17] B. Weir, *Genetic Data analysis II*, vol. 2. Sinauer Associates, Inc., 1996.

[18] B. Manly, *Randomization, Bootstrap and Monte Carlo Methods in Biology.* Chapman & Hall, London, UK, 1997.

[19] N. Rosenberg, L. Li, R. Ward, and J. Pritchard, “Informativeness of genetic markers for inference of ancestry.,” *American Journal of Human Genetics*, vol. 73, no. 6, pp. 1402–22, 2003.

[20] M. Whitlock and D. McCauley, “Indirect measures of gene flow and migration: FST not equal to 1/(4Nm + 1).,” *Heredity*, vol. 82 (Pt 2), pp. 117–25, Feb. 1999.

[21] P. O’Reilly, M. Canino, K. Bailey, and P. Bentzen, “Inverse relationship between F and microsatellite polymorphism in the marine fish, walleye pollock (Theragra chalcogramma): implications for resolving weak population structure.,” *Molecular ecology*, vol. 13, pp. 1799–814, July 2004.

[22] F. Rousset, “genepop’007: a complete re-implementation of the genepop software for Windows and Linux.,” *Molecular ecology resources*, vol. 8, no. 1, pp. 103–6, 2008.