



diveRsity v1.2.3 Help Manual

(compiled version)

by *Kevin Keenan*

kevinkeen02@qub.ac.uk

October 5, 2012

Contents

1	Introduction	2
1.1	About R	2
1.2	About <code>diveRsity</code> v1.2.3	2
1.2.1	What's new in version 1.2.3?	3
2	Setup	4
2.1	Installing R	4
2.2	Installing <code>diveRsity</code>	4
2.3	Installing optional enhancer packages	4
2.4	Loading <code>diveRsity</code>	4
3	Function details	5
3.1	<code>div.part()</code>	5
3.1.1	Standard formulae	5
3.1.2	Estimator formulae	6
3.1.3	Bootstrapping	7
3.2	<code>in.calc()</code>	8
3.3	<code>readGenepop.user()</code>	8
4	Function Usage	9
4.1	<code>div.part()</code>	9
4.1.1	Arguments	9
4.1.2	Returned values	11
4.2	<code>in.calc()</code>	22
4.2.1	Arguments	22
4.2.2	Returned values	24
4.3	<code>readGenepop.user()</code>	28
4.3.1	Arguments	28
4.3.2	Returned values	28
5	Examples	30
5.1	<code>div.part</code>	30
5.1.1	Setting your <code>working directory</code>	30
5.1.2	Loading <code>Test_data</code>	30
5.1.3	Running <code>div.part</code>	31
5.1.4	Accessing your results within the R session	31
5.2	<code>in.calc</code>	33
5.2.1	Setting your <code>working directory</code>	33
5.2.2	Loading <code>Test_data</code>	33
5.2.3	Running <code>in.calc</code>	34
5.2.4	Accessing your results within the R session	34
5.3	<code>readGenepop.user</code>	36
5.3.1	Setting your <code>working directory</code>	36
5.3.2	Loading <code>Test_data</code>	36
5.3.3	Running <code>readGenepop.user</code>	36
5.3.4	Accessing your results within the R session	37
5.3.5	Applications for <code>readGenepop.user</code>	37
5.3.6	Using <code>readGenepop.user</code> to bootstrap the number of alleles per locus	39

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 the suggested packages **xlsx** and **sendplot**. Fully reproducible Worked examples for each function will be provide as a guide to how the package should be used. 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/useRs. More information about **R** can be found at <http://www.r-project.org/about.html>.

1.2 About diveRsity v1.2.3

diveRsity is a package containing three 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** makes optional use of two additional **R** packages, **xlsx** and **sendplot**, for returning results to useRs. Their installation is explained below.

diveRsity was written to ensure that even **R** beginners could carry out analyses 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 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 in version 1.2.3?

Version 1.2.0 and up introduces a complete rewrite of `diveRsity v1.0`, which, after publication on CRAN, it was realised, was written almost entirely “in a C accent” (to quote the *R Inferno*). The original code contained extensive uses of nested loops rather than vectorized functions. Version 1.2.0 and up has been vectorized in all but the least computationally intensive pieces of code.

Parallel computations are also now possible 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 in the function `div.part`. 95% confidence intervals are also calculated for the F-statistics.

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 analysis time. For this reason, the calculation of F-statistics has been included as an optional extra through the argument `WC_Fst`.

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), type the following command:

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

If this is the first package you have installed you will be prompted to select a CRAN mirror. Choose one which is preferably in your country or as close to your country as possible. Providing you have chosen a mirror the package will download and install.

2.3 Installing optional enhancer packages

The `xlsx` and `sendplot` packages are optional for the purposes of writing and plotting analysis results from the `div.part` and `in.calc`. For parallel computations, the packages `doSNOW` (Linux) or `doParallel` (Windows), and their dependencies are required. Should you wish to use employ any of these packages use for example:

```
install.packages(c("xlsx", "sendplot"))  
install.packages("doSNOW")
```

N.B. You could reasonably install all packages together using:

```
install.packages(c("diveRsity", "xlsx", "sendplot", "doSNOW"))
```

2.4 Loading `diveRsity`

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

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

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} [2, 3]

$$G_{st} = \frac{D_{st}}{H_t} \quad (1)$$

Where $D_{st} = H_t - H_s$, H_t is the total heterozygosity and H_s is intra-population heterozygosity.

G'_{st} [4]

$$G'_{st} = \frac{G_{st}}{G_{st(max)}} \quad (2)$$

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} [5]

$$D_{Jost} = \left[\frac{(H_t - H_s)}{(1 - H_s)} \right] \left[\frac{n}{(n - 1)} \right] \quad (3)$$

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 [6] (eqn 6 below). The formulae are as follows:

\hat{H}_s [3]

$$\hat{H}_s = H_s \left[\frac{2\bar{N}}{(2\bar{N} - 1)} \right] \quad (4)$$

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

\hat{H}_t [3]

$$\hat{H}_t = H_t + \left[\frac{\hat{H}_s}{(2\bar{N}n)} \right] \quad (5)$$

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.

$D_{est(Chao)}$ [6, 5]

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

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. $\hat{\theta}$) [7, 8]

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

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

Sampling variance each statistic can be assessed using the bootstrapping method implemented in **diveRsity**. 95% confidence intervals are calculated using the method described in [9].

3.2 `in.calc()`

`in.calc` allows the calculation of locus informativeness for 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:

$I_n(\text{alleles})$ [10]

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

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.

$I_n(\text{locus})$ [10]

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

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. The package returns up to 17 `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`.

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

<code>infile</code>	Specifying the name of the ‘ <i>genepop</i> ’ [11] file from which the statistics are to be calculated. This file can be in either the 3 digit or 2 digit format, and must contain only one whitespace separator (e.g. “space” or “tab”) between each column including the individual names column. The name must be a character string.
<code>outfile</code>	Allows users to specify a prefix for an output folder. Name must be a character string enclosed in either “” or ‘’.
<code>gp</code>	Specifies the digit format of the <code>infile</code> . Either 3 (default) or 2.
<code>WC_Fst</code>	A logical indication as to whether Weir and Cockerham’s, 1984 F-statistics should be calculated. This option will increase analysis time.
<code>bs_locus</code>	Gives users the option to bootstrap locus statistics. Results will be written to <i>.xlsx</i> workbook by default if the package <code>xlsx</code> is installed, and to a <i>.html</i> file if <code>Plot=TRUE</code> . If <code>xlsx</code> is not installed, results will be written to <i>.txt</i> files.
<code>bs_pairwise</code>	Gives users the option to bootstrap statistics across all loci for each pairwise population comparison. Results will be written to a <i>.xlsx</i> file by default if the package <code>xlsx</code> is installed, and to a <i>.html</i> file if <code>Plot=TRUE</code> . If <code>xlsx</code> is not installed, results will be written to <i>.txt</i> files.

Arguments cont.

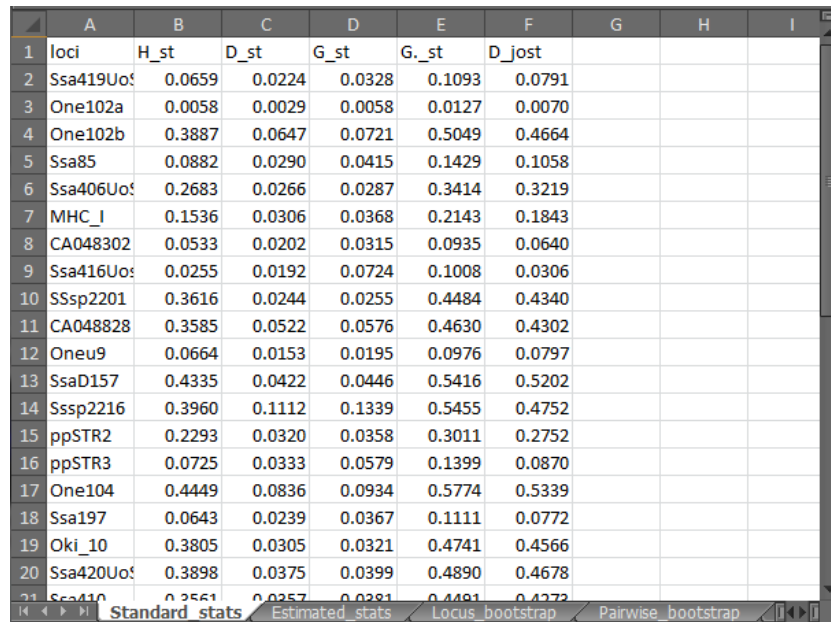
<code>bootstraps</code>	Determines the number of bootstrap iterations to be carried out. The default value is <code>bootstraps = 0</code> , this is only valid when all bootstrap options are false. There is no upper limit on the number of bootstrap iterations, however very large numbers of bootstrap iterations for pairwise calculations (> 1000) may take a long time to run for large data sets. 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.
<code>Plot</code>	Optional interactive <i>.html</i> image files of the plotted bootstrap results for loci if <code>bs_locus = TRUE</code> and pairwise population comparisons if <code>bs_pairwise = TRUE</code> and the package <code>sendplot</code> is installed. The default option is <code>Plot = FALSE</code> .
<code>parallel</code>	A logical argument specifying if computations should be run in parallel on all available CPU cores. If <code>parallel = TRUE</code> , batches of jobs will be distributed to all cores resulting in faster completion. In Windows, the packages <code>doParallel</code> , <code>iterators</code> , <code>parallel</code> (distributed with R) and <code>foreach</code> must be installed to use parallel computation. In Linux the packages <code>doSNOW</code> , <code>parallel</code> , <code>snow</code> , <code>iterators</code> and <code>foreach</code> 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 two and four depending on the argument options chosen.

An example screenshot of the `.xlsx` output file is shown below:

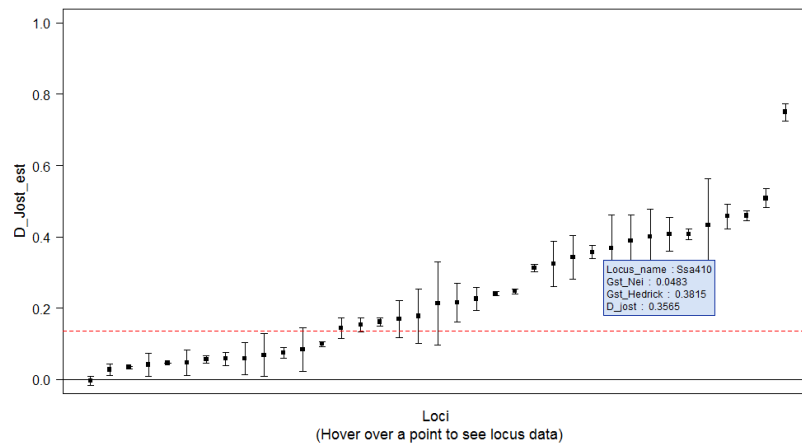


	A	B	C	D	E	F	G	H	I
1	loci	H_st	D_st	G_st	G_st	D_jost			
2	Ssa419Uo	0.0659	0.0224	0.0328	0.1093	0.0791			
3	One102a	0.0058	0.0029	0.0058	0.0127	0.0070			
4	One102b	0.3887	0.0647	0.0721	0.5049	0.4664			
5	Ssa85	0.0882	0.0290	0.0415	0.1429	0.1058			
6	Ssa406Uo	0.2683	0.0266	0.0287	0.3414	0.3219			
7	MHC_I	0.1536	0.0306	0.0368	0.2143	0.1843			
8	CA048302	0.0533	0.0202	0.0315	0.0935	0.0640			
9	Ssa416Uo	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	Ok1_10	0.3805	0.0305	0.0321	0.4741	0.4566			
20	Ssa420Uo	0.3898	0.0375	0.0399	0.4890	0.4678			
21	Ssa410	0.2561	0.0257	0.0281	0.4481	0.4272			

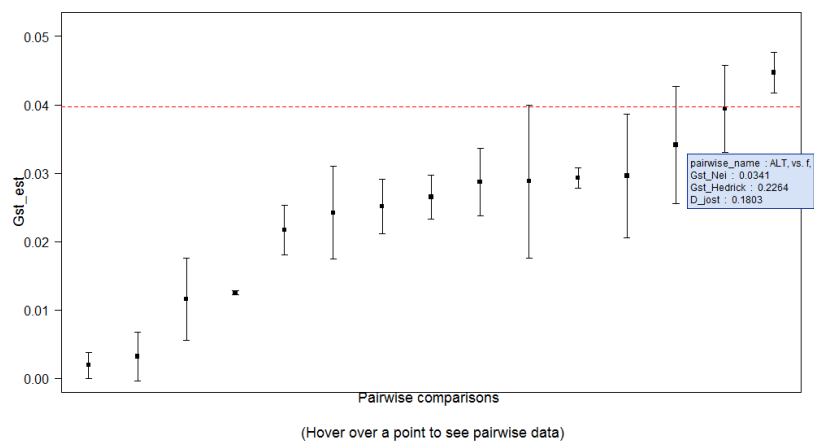
Returned values cont.

Examples of the interactive plots written, if `xlsx` is available, are given below. Error bars represent bootstrapped 95% ci's.

Example of bootstrapped locus results plot



Example of bootstrapped pairwise results plot



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 sample and loci.

	loci	H_st	D_st	G_st	G_hed_st	D_jost
[1,]	Locus1	0.0659	0.0224	0.0328	0.1093	0.0791
[2,]	Locus2	0.0058	0.0029	0.0058	0.0127	0.007
[3,]	Locus3	0.3887	0.0647	0.0721	0.5049	0.4664
[4,]	Locus4	0.0882	0.029	0.0415	0.1429	0.1058
[5,]	Locus5	0.2683	0.0266	0.0287	0.3414	0.3219
[6,]	Locus6	0.1536	0.0306	0.0368	0.2143	0.1843
[7,]	Locus7	0.0533	0.0202	0.0315	0.0935	0.064
[8,]	Locus8	0.0255	0.0192	0.0724	0.1008	0.0306
[9,]	Locus9	0.3616	0.0244	0.0255	0.4484	0.434
[10,]	Locus10	0.3585	0.0522	0.0576	0.463	0.4302
[11,]	Global			0.0493	0.2163	0.1757

loci

A list of locus names

H_st

Between subpopulation heterozygosity per locus

D_st

Absolute differentiation per locus [2]

G_st

F_st analogue for multiple alleles per locus [2]

G_hed_st

Hedrick's standardized "differentiation" per locus [4]

D_jost

Jost's true allelic differentiation per locus [5]

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 sample and loci.

	loci	Harmonic_N	H_st_est	D_st_est	G_st_est	G_hed_st_est	D_Jost_est
Locus1	Locus1	43.1218	0.6841	0.016	0.0234	0.0799	0.0578
Locus2	Locus2	43.5209	0.5035	-0.0019	-0.0038	-0.0084	-0.0046
Locus3	Locus3	43.6403	0.8998	0.0566	0.0629	0.4688	0.4332
Locus4	Locus4	43.4476	0.7012	0.0225	0.0321	0.1134	0.084
Locus5	Locus5	42.7674	0.9291	0.0177	0.0191	0.2542	0.2397
Locus6	Locus6	43.4476	0.8329	0.0228	0.0274	0.1675	0.1441
Locus7	Locus7	43.4476	0.6429	0.0142	0.0221	0.067	0.0459
Locus8	Locus8	43.2566	0.2657	0.0168	0.0632	0.0884	0.0268
Locus9	Locus9	43.0673	0.9587	0.0153	0.016	0.3352	0.3244
Locus10	Locus10	43.2469	0.9083	0.0439	0.0483	0.4181	0.3885
All	Global				0.0397	0.1806	0.1462
	Fis_WC	Fst_WC	Fit_WC				
Locus1	0.0363	0.0257	0.061				
Locus2	-0.0474	-0.0042	-0.0518				
Locus3	0.0266	0.0745	0.0991				
Locus4	0.0205	0.0357	0.0555				
Locus5	0.0539	0.0222	0.0749				
Locus6	0.201	0.03	0.225				
Locus7	0.0173	0.0258	0.0427				
Locus8	0.1976	0.0689	0.2529				
Locus9	0.0407	0.0189	0.0588				
Locus10	0.0448	0.0564	0.0986				
All	0.0655	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 [3]

D_st_est

Estimator of absolute differentiation [3]

G_st_est

Nearly unbiased estimator of G_{st} [3]

G_hed_st_est

Estimator of Hedrick's G'_{st} [4]

D_Jost_est

Estimator of Jost's D [5]

Fis_WC

Weir and Cockerham's inbreeding coefficient estimator [7]

Fst_WC

Weir and Cockerham's fixation index estimator [7]

Fit_WC

Weir and Cockerham's overall fixation index estimator [7]

Returned values cont.

\$pairwise A list of six matrices containing pairwise diversity statistics without bootstrapped confidence intervals.

[1] Gst

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

[1] G_hed_st

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

[1] D_Jost

```
      pop1, pop2, pop3, pop4,
pop1, --
pop2, 0.0409 --
pop3, 0.2254 0.2011 --
pop4, 0.1989 0.179 0.0519 --
```

[1] Gst_est

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

[1] G_hed_st_est

```
      pop1, pop2, pop3, pop4,
pop1, --
pop2, 0.0124 --
pop3, 0.2264 0.1954 --
pop4, 0.1992 0.1732 0.0224 --
```

[1] D_Jost_est

```
      pop1, pop2, pop3, pop4,
pop1, --
pop2, 0.0027 --
pop3, 0.1803 0.1579 --
pop4, 0.1484 0.1325 0.0102 --
```

```

[1] Fis_WC

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

[1] Fst_WC

      pop1, pop2, pop3, pop4,
pop1, --
pop2, 0.0027 --
pop3, 0.0647 0.0543 --
pop4, 0.0563 0.0478 0.0057 --

[1] Fit_WC

      pop1, pop2, pop3, pop4,
pop1, --
pop2, 0.0933 --
pop3, 0.1323 0.1331 --
pop4, 0.1233 0.1245 0.0689 --

```

Returned values cont.

`$bs_locus` A list containing six matrices of locus values for G_{st} , G'_{st} , D_{Jost} , $G_{st(est)}$, $G'_{st(est)}$, and $D_{Jost(est)}$ along with their respective 95% confidence interval.

[1] Gst

	Actual	Lower_CI	Upper_CI
Locus1	0.0328	0.0076	0.0580
Locus2	0.0058	-0.0133	0.0249
Locus3	0.0721	0.0584	0.0858
global	0.0493	0.0447	0.0539

[1] G_hed_st

	Actual	Lower_CI	Upper_CI
Locus1	0.1093	0.0347	0.1839
Locus2	0.0127	-0.0279	0.0533
Locus3	0.5049	0.4531	0.5567
global	0.2163	0.2013	0.2313

[1] D_Jost

	Actual	Lower_CI	Upper_CI
Locus1	0.0791	0.0233	0.1349
Locus2	0.0070	-0.0152	0.0292
Locus3	0.4664	0.4164	0.5164
global	0.1757	0.1635	0.1879

[1] Gst_est

	Actual	Lower_CI	Upper_CI
Locus1	0.0234	-0.0018	0.0486
Locus2	-0.0038	-0.0231	0.0155
Locus3	0.0629	0.0491	0.0767
global	0.0397	0.0350	0.0444

[1] G_hed_st_est

	Actual	Lower_CI	Upper_CI
Locus1	0.0799	0.0027	0.1571
Locus2	-0.0084	-0.0503	0.0335
Locus3	0.4688	0.4119	0.5257
global	0.1806	0.1642	0.1970

[1] D_Jost_est

	Actual	Lower_CI	Upper_CI
Locus1	0.0578	0.0004	0.1152
Locus2	-0.0046	-0.0275	0.0183
Locus3	0.4332	0.3788	0.4876
global	0.1462	0.1291	0.1633

```
[1] Fis_WC
```

	[,1]	[,2]	[,3]
[1,]	0.0363	-0.0669	0.1395
[2,]	-0.0474	-0.1467	0.0519
[3,]	0.0266	-0.0267	0.0799
[4,]	0.0655	0.0510	0.0800

```
[1] Fst_WC
```

	[,1]	[,2]	[,3]
[1,]	0.0257	-0.0026	0.0540
[2,]	-0.0042	-0.0269	0.0185
[3,]	0.0745	0.0583	0.0907
[4,]	0.0456	0.0402	0.0510

```
[1] Fit_WC
```

	[,1]	[,2]	[,3]
[1,]	0.0610	-0.0281	0.1501
[2,]	-0.0518	-0.1342	0.0306
[3,]	0.0991	0.0536	0.1446
[4,]	0.1081	0.0957	0.1205

Returned values cont.

\$bs_pairwise A list containing six matrices of pairwise values for G_{st} , G'_{st} , D_{Jost} , $G_{st(est)}$, $G'_{st(est)}$, and $D_{Jost(est)}$ along with their respective 95% confidence interval.

[1] Gst

	Actual	Lower_CI	Upper_CI
pop1, vs. pop2,	0.0077	0.0052	0.0102
pop1, vs. pop3,	0.0401	0.0350	0.0452
pop1, vs. pop4,	0.0349	0.0311	0.0387
pop5, vs. pop6,	0.0281	0.0218	0.0344

[1] G_hed_st

	Actual	Lower_CI	Upper_CI
pop1, vs. pop2,	0.0486	0.0339	0.0633
pop1, vs. pop3,	0.2562	0.2291	0.2833
pop1, vs. pop4,	0.2271	0.2090	0.2452
pop5, vs. pop6,	0.1943	0.1613	0.2273

[1] D_Jost

	Actual	Lower_CI	Upper_CI
pop1, vs. pop2,	0.0409	0.0284	0.0534
pop1, vs. pop3,	0.2254	0.2007	0.2501
pop1, vs. pop4,	0.1989	0.1829	0.2149
pop5, vs. pop6,	0.1710	0.1419	0.2001

[1] Gst_est

	Actual	Lower_CI	Upper_CI
pop1, vs. pop2,	0.0019	-0.0007	0.0045
pop1, vs. pop3,	0.0341	0.0290	0.0392
pop1, vs. pop4,	0.0296	0.0258	0.0334
pop5, vs. pop6,	0.0217	0.0154	0.0280

[1] G_hed_st_est

	Actual	Lower_CI	Upper_CI
pop1, vs. pop2,	0.0124	-0.0037	0.0285
pop1, vs. pop3,	0.2264	0.1986	0.2542
pop1, vs. pop4,	0.1992	0.1798	0.2186
pop5, vs. pop6,	0.1568	0.1216	0.1920

[1] D_Jost_est

	Actual	Lower_CI	Upper_CI
pop1, vs. pop2,	0.0027	-0.0134	0.0188
pop1, vs. pop3,	0.1803	0.1411	0.2195
pop1, vs. pop4,	0.1484	0.1249	0.1719
pop5, vs. pop6,	0.1199	0.0916	0.1482

[1] Fis_WC

	Actual	Lower_CI	Upper_CI
pop1, vs. pop2,	0.0908	0.0673	0.1143
pop1, vs. pop3,	0.0723	0.0412	0.1034
pop1, vs. pop4,	0.0711	0.0542	0.0880
pop5, vs. pop6,	0.0420	0.0273	0.0567

[1] Fst_WC

	Actual	Lower_CI	Upper_CI
pop1, vs. pop2,	0.0027	-0.0025	0.0079
pop1, vs. pop3,	0.0647	0.0553	0.0741
pop1, vs. pop4,	0.0563	0.0492	0.0634
pop5, vs. pop6,	0.0417	0.0298	0.0536

[1] Fit_WC

	Actual	Lower_CI	Upper_CI
pop1, vs. pop2,	0.0933	0.0720	0.1146
pop1, vs. pop3,	0.1323	0.1028	0.1618
pop1, vs. pop4,	0.1233	0.1098	0.1368
pop5, vs. pop6,	0.0820	0.0739	0.0901

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 ‘ <i>genepop</i> ’ [11] file from which the statistics are to be calculated. This file can be in either the 3 digit or 2 digit format, and must contain only one whitespace separator (e.g. “space” or “tab”) between each column including the individual names column. The name must be a character string.
outfile	Allows users to specify a suffix for output folder and files. Name must be 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 <i>.xlsx</i> file by default if the package xlsx is installed, and to a <i>.png</i> file if Plot=TRUE . If xlsx is not installed, results will be written to <i>.txt</i> files.
bs_pairwise	Gives users the option to bootstrap statistics across all loci for each pairwise population comparison. Results will be written to a <i>.xlsx</i> file by default if the package xlsx is installed. If xlsx is not installed, results will be written to <i>.txt</i> files.

Arguments cont.

<code>bootstraps</code>	Determines the number of bootstrap iterations to be carried out. The default value is <code>bootstraps = 0</code> , 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.
<code>Plot</code>	Optional <code>.png</code> image file of the plotted bootstrap results for locus I_n if <code>bs_locus = TRUE</code> . The default option is <code>Plot = FALSE</code> .
<code>parallel</code>	A logical argument specifying if computations should be run in parallel on all available CPU cores. If <code>parallel = TRUE</code> , batches of jobs will be distributed to all cores resulting in faster completion. In Windows, the packages <code>doParallel</code> , <code>iterators</code> , <code>parallel</code> (distributed with R) and <code>foreach</code> must be installed to use parallel computation. In Linux the packages <code>doSNOW</code> , <code>parallel</code> , <code>snow</code> , <code>iterators</code> and <code>foreach</code> should be installed.

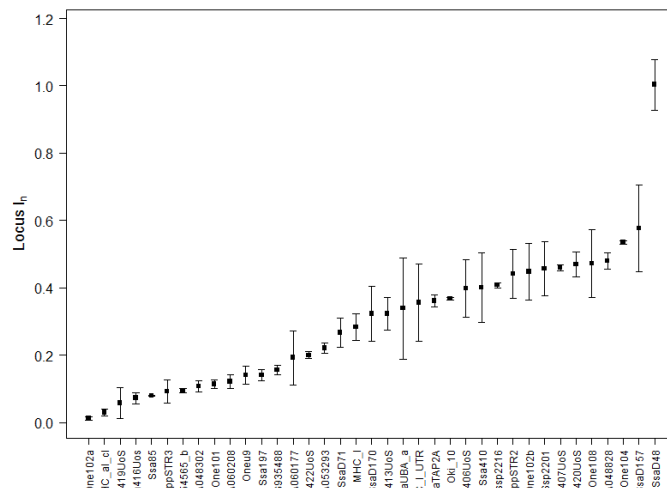
4.2.2 Returned values

Values returned from `in.calc` are a single `.xlsx` workbook (if `xlsx` is available), 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

	A	B	C	D	E	F	G	H
1	LocI	Actual_In	Lower_95I	Upper_95CI				
2	ALT, vs. LG,							
3	Ssa419UoS	0.0234	0.018	0.0288				
4	One102a	0.0131	-0.0013	0.0275				
5	One102b	0.0794	-0.0534	0.2122				
6	Ssa85	0.0205	0.008	0.033				
7	Ssa406UoS	0.0578	0.0221	0.0935				
8	MHC_I	0.0541	0.0016	0.1066				
9	CA048302	0.0146	3.00E-04	0.0289				
10	Ssa416UoS	0.0014	-0.0047	0.0075				
11	Sssp2201	0.1554	0.0745	0.2363				
12	CA048828	0.0472	0.0119	0.0825				
13	Oneu9	0.0431	-3.00E-04	0.0865				
14	SsaD157	0.0804	0.0052	0.1556				
15	Sssp2216	0.0059	-0.0334	0.0452				
16	ppSTR2	0.1287	0.1283	0.1291				
17	ppSTR3	0.0237	-0.0044	0.0518				
18	One104	0.056	0.0226	0.0894				
19	Ssa197	0.0244	0.0244	0.0244				
20	Ok_i	0.0889	0.0885	0.0893				
21	Ssa420UoS	0.084	0.0314	0.1366				
22	Ssa410	0.1169	0.055	0.1788				
23	BG935488	0.03	-0.019	0.079				
24	SsaD71	0.0271	-0.1049	0.1591				
25	SasaTAP2A	0.0228	0.0096	0.036				
26	CA053283	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 `infile`. Each column represents the allele specific I_n per locus except the last column, which contains the locus I_n for each locus.

Returned values cont.

l_bootstrap A character matrix of locus I_n 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	0.0247	0.0915
Locus2	0.0123	0.0010	0.0236
Locus3	0.4482	0.3814	0.5150
Locus4	0.0800	0.0344	0.1256
Locus5	0.3983	0.3274	0.4692
Locus6	0.2839	0.2258	0.3420
Locus7	0.1068	0.0656	0.1480
Locus8	0.0728	0.0456	0.1000
Locus9	0.4571	0.4080	0.5062
Locus10	0.4799	0.4290	0.5308

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	-0.0109	0.0577
Locus2	0.0131	-0.0039	0.0301
Locus3	0.0794	0.0423	0.1165
Locus4	0.0205	-0.0284	0.0694
Locus5	0.0578	0.0211	0.0945

[1] pop1, vs. pop3,

	In	Lower_95CI	Upper_95CI
Locus1	0.0167	-0.0049	0.0383
Locus2	0.0115	-0.0088	0.0318
Locus3	0.3157	0.2233	0.4081
Locus4	0.0982	0.0397	0.1567
Locus5	0.2427	0.1860	0.2994

[1] pop1, vs. pop4,

	In	Lower_95CI	Upper_95CI
Locus1	0.0233	-0.0047	0.0513
Locus2	0.0112	0.0002	0.0222
Locus3	0.3395	0.2793	0.3997
Locus4	0.0419	0.0064	0.0774
Locus5	0.2794	0.2112	0.3476

[1] pop1, vs. pop5,

	In	Lower_95CI	Upper_95CI
Locus1	0.0619	0.0307	0.0931
Locus2	0.0118	-0.0011	0.0247
Locus3	0.3690	0.2742	0.4638
Locus4	0.0630	0.0148	0.1112
Locus5	0.2615	0.1950	0.3280

[1] pop1, vs. pop6,

	In	Lower_95CI	Upper_95CI
Locus1	0.0264	0.0016	0.0512
Locus2	0.0123	0.0008	0.0238
Locus3	0.2815	0.2376	0.3254
Locus4	0.0297	-0.0097	0.0691
Locus5	0.2187	0.1411	0.2963

4.3 readGenepop.user()

The general usage of `readGenepop.user` is:

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

4.3.1 Arguments

<code>infile</code>	Specifying the name of the ' <i>genepop</i> ' file from which the statistics are to be calculated. This file can be in either the 3 digit or 2 digit format, and must contain only one <i>whitespace</i> separator (e.g. "space" or "tab") between each column including the individual names column (i.e. no whitespace between the name and comma). The number of columns must be equal to the number of loci + 1 (the individual names column). If this file is not in the working directory the file path must be given. The name must be a character string (i.e. enclosed in "" or ").
<code>gp</code>	Specifies the digit format of the <code>infile</code> . Either 3 (default) or 2.
<code>bootstrap</code>	A logical argument indicating whether the <code>infile</code> should be sampled with replacement. All other values are returned as normal if <code>bootstrap = TRUE</code> , however an additional object, <code>bs_file</code> is also returned.

4.3.2 Returned values

<code>npops</code>	The number of population samples in <code>infile</code> .
<code>nloci</code>	The number of loci in <code>infile</code> .
<code>pop_alleles</code>	A list of matrices ($n = 2 \times \text{npops}$) containing haploid allele designations. Every two matrices contain the two alleles per individual per population. For example <code>pop_alleles[[1]][1,1]</code> and <code>pop_alleles[[2]][1,1]</code> are the two alleles observed in individual '1' in population '1' at locus '1', whereas <code>pop_alleles[[3]][1,1]</code> and <code>pop_alleles[[4]][1,1]</code> are the two alleles observed in individual '1' in population '2' at locus '1'.
<code>pop_list</code>	A list of matrices ($n = \text{npops}$) containing the diploid genotypes of individuals per locus.
<code>loci_names</code>	A character vector containing the names of loci from <code>infile</code> .
<code>pop_pos</code>	A numeric vector or the row index locations of the first individual per population in <code>infile</code> .
<code>pop_sizes</code>	A numeric vector of length <code>npops</code> containing the number of individuals per population sample in <code>infile</code> .
<code>allele_names</code>	A list of <code>npops</code> lists containing <code>nloci</code> character vectors of alleles names per locus. Useful for identifying unique alleles.
<code>all_alleles</code>	A list of <code>nloci</code> character vectors of all alleles observed across all population samples in <code>infile</code> .

<code>allele_freq</code>	A list containing <code>nloci</code> matrices containing allele frequencies per alleles per population sample.
<code>raw_data</code>	An unaltered data frame of <code>infile</code> .
<code>loci_harm_N</code>	A numeric vector of length <code>nloci</code> , containing the harmonic mean number of individuals genotyped per locus.
<code>n_harmonic</code>	A numeric value representing the harmonic mean of <code>npops</code> .
<code>pop_names</code>	A character vector containing a six character population sample name for each population in <code>infile</code> (the first six characters of the first individual).
<code>indtyp</code>	A list of length <code>nloci</code> containing character vectors of length <code>npops</code> , indicating the number of individuals per population sample typed at each locus.
<code>nalleles</code>	A vector of the total number of alleles observed at each locus.
	A list of matrices of the observed number of allele occurrences per population.

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 taken 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 = 100,
+                          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"
```


Accessing your results within the R session cont.

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 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.

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	0.0130	0.0526
Locus2	0.0058	-0.0099	0.0215
Locus3	0.0721	0.0583	0.0859
Locus4	0.0415	0.0187	0.0643
Locus5	0.0287	0.0208	0.0366
Locus6	0.0368	0.0226	0.0510
Locus7	0.0315	0.0130	0.0500
Locus8	0.0724	0.0259	0.1189
Locus9	0.0255	0.0175	0.0335
Locus10	0.0576	0.0419	0.0733

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 = 100,
+                         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_bootstrap`, 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	-0.0073	0.0541
Locus2	0.0131	0.0001	0.0261
Locus3	0.0794	0.0200	0.1388

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"       "ls"             "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.

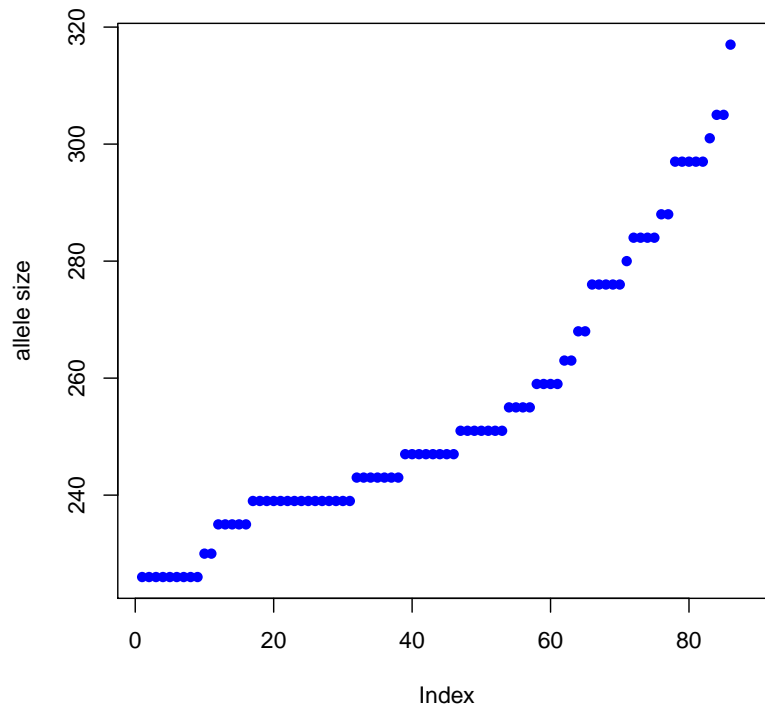
‘*Ad hoc*’ investigation of locus mutation model

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* \times 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 if allele fragments fit 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 Using readGenepop.user to bootstrap the number of alleles per locus

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 as 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	6.5	5.466989	7.533011
2	2.9	2.280194	3.519806
3	17.8	16.973591	18.626409
4	7.4	6.029556	8.770444
5	34.8	31.626118	37.973882

6	13.9	13.280194	14.519806
7	8.6	7.587860	9.612140
8	4.0	4.000000	4.000000
9	41.2	38.459113	43.940887
10	33.4	31.099376	35.700624

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] 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.
- [3] M. Nei and R. Chesser, “Estimation of fixation indices and gene diversities,” *Ann. Hum. Genet.*, vol. 47, no. Pt 3, pp. 253–259, 1983.
- [4] P. Hedrick, “A standardized genetic differentiation measure,” *Evolution*, vol. 59, no. 8, pp. 1633–1638, 2005.
- [5] L. Jost, “G ST and its relatives do not measure differentiation,” *Molecular Ecology*, vol. 17, no. 18, pp. 4015–4026, 2008.
- [6] 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.
- [7] B. Weir and C. Cockerham, “Estimating F-statistics for the analysis of population structure,” *Evolution*, vol. 38, no. 6, pp. 1358–1370, 1984.
- [8] B. Weir, *Genetic Data analysis II*, vol. 2. Sinauer Associates, Inc., 1996.
- [9] B. Manly, *Randomization, Bootstrap and Monte Carlo Methods in Biology*. Chapman & Hall, London, UK, 1997.
- [10] 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.
- [11] 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.