The diveRsity Package

Kevin Keenan

Saturday, December 20, 2014

1 Introduction

This document is in progress.

2 Function demonstrations

2.1 arp2gen

This is a function allowing users to convert *arliquin* genotype files to *genepop* files. The digit format of the resulting genepop file is determined my the genotype format in the *arlequin* file. The name of the genepop file created is equal to the name of the arlequin file with the replacement of the **.arp** file extension with **.gen**. Missing data in the *arlequin* file must be indicated by '-9', and will be converted to '00' or '000' in the genepop file, depending on the allele format in the *arlequin* file.

```
arp2gen(infile)
```

2.1.0.1 General use

2.1.0.2 Argument description

- `infile`: A character string pointing to an *arlequin* file to be converted. This string can either be a file name, providing the file is in the current working directory, otherwise an absolute or relative file path can be provided within the string.

2.1.0.3 Example Here an input file named *arp2gen.arp* in the current working directory has the following format:

```
#Arlequin input file written by the simulation program fastsimcoal.exe

[Profile]
    Title="A series of simulated samples"
    NbSamples=2

    GenotypicData=1
    GameticPhase=0
    RecessiveData=0
    DataType=MICROSAT
    LocusSeparator=WHITESPACE
    MissingData='?'

[Data]
```

```
    [[Samples]]

#Number of independent chromosomes: 1
#Total number of polymorphic sites: 5
# 10 polymorphic positions on chromosome 1
#1, 2, 3, 4, 5

        SampleName="Sample 1"
        SampleSize=5
        SampleData= {
1_1 1    501  500  500  501  499
         501  500  500  500  499
1_2 1    498  500  500  501  500
         501  501  500  500  499
1_3 1    502  500  500  501  498
         501  500  500  500  499
1_4 1    502  500  500  501  498
         501  500  500  501  499
1_5 1    500  500  499  501  499
         499  501  500  501  500

}
        SampleName="Sample 2"
        SampleSize=5
        SampleData= {
2_1 1    501  500  500  501  499
         501  500  500  501  499
2_2 1    501  500  500  501  499
         501  500  500  501  499
2_3 1    501  500  500  501  499
         498  500  501  501  500
2_4 1    501  500  500  501  499
         501  500  500  501  499
2_5 1    501  500  500  501  499
         500  500  500  501  499

}

[[Structure]]

    StructureName="Simulated data"
    NbGroups=1
    Group={
       "Sample 1"
       "Sample 2"
    }
```

This file can be converted to *genepop* format as follows:

```
arp2gen("arp2gen.arp")
```

This command results in a *genepop* file named _arp2gen.gen with the following format being written to the current working directory:

```
./manuscript/arp2gen/arp2gen_gen_converted
locus1
locus2
locus3
locus4
locus5
POP
pop1 ,  501501  500500  500500  501500  499499
pop1 ,  498501  500501  500500  501500  500499
pop1 ,  502501  500500  500500  501500  498499
pop1 ,  502501  500500  500500  501501  498499
pop1 ,  500499  500501  499500  501501  499500
POP
pop2 ,  501501  500500  500500  501501  499499
pop2 ,  501501  500500  500500  501501  499499
pop2 ,  501498  500500  500501  501501  499500
pop2 ,  501501  500500  500500  501501  499499
pop2 ,  501500  500500  500500  501501  499499
```

2.2 chiCalc

This function allows users to test for sample independence from genotype counts. It implements Fisher's exact tests using the `fisher.test` function. Both global and pairwise tests are available, and can be tested across loci and per locus. Multilocus p values are calculated using Fisher's method.

```
chiCalc(infile = NULL, outfile = NULL, pairwise = TRUE, mcRep = 2000)
```

2.2.0.4 General use

2.2.0.5 Argument description

- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format.
- `outfile`: A character string indicating the prefix to be added to the results directory created. All results files will be written to this directory.
- `pairwise`: A logical argument indicating whether sample independence should be tested between all population pairs.
- `mcRep`: An integer specifying the number Monte Carlo test replicates. See ?fisher.test for more information

2.2.0.6 Example In the file `Test_data`, there are six population samples. To test sample independence between all pairs if populations, the following code is used.

```r
# load diveRsity
library(diveRsity)
# load test data
data(Test_data)
# calculate pairwise tests
res <- chiCalc(infile = Test_data, pairwise = TRUE)
head(res$multilocus_pw)
```

```
            pops p.value
1 pop1, vs pop2,  0.7394
2 pop1, vs pop3,  0.0000
3 pop1, vs pop4,  0.0000
4 pop1, vs pop5,  0.0000
5 pop1, vs pop6,  0.0000
6 pop2, vs pop3,  0.0000
```

These results indicate that there is a significant difference in the distribution of genotypes between all population pairs shown (i.e. $p < 0.05$), except between pop1 and pop2.

2.3 corPlot

Downward biases in the estimation of F_{ST} like parameters as a result of locus polymorphism are well described. This function allows users to visually explore the effect of this bias within their data by comparing the relationship between polymorphism (mean number of alleles per locus) and differentiation (calculated for F_{ST}, G_{ST}, G'_{ST} and D_{Jost}). The general rule for this method is, if there is a negative relationship between number of alleles and either F_{ST} or G_{ST}, but a neutral or positive relationship between number of alleles and G'_{ST} or D_{Jost}, then F_{ST} type estimates are likely to be biased, and the use of D_{Jost} is recommended for measuring genetic differentiation.

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

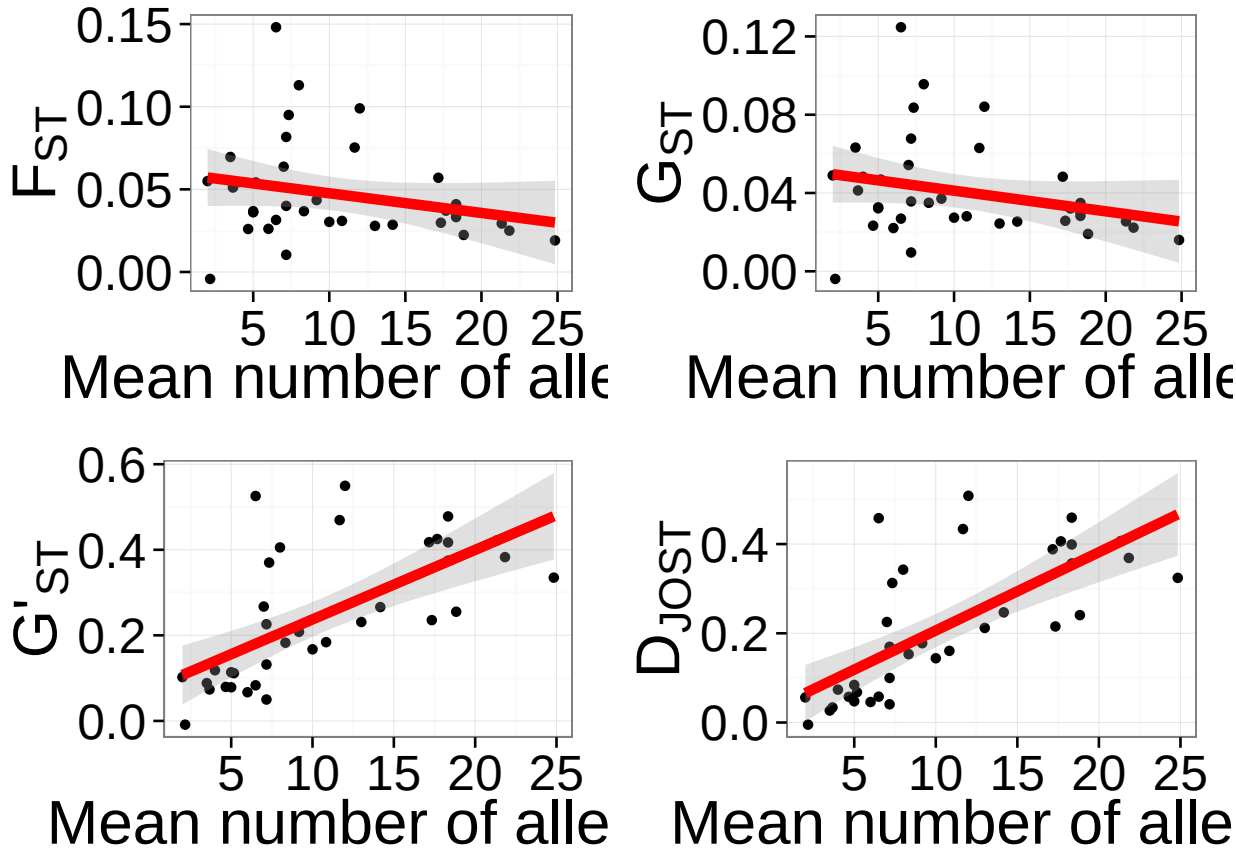
2.3.0.7 General use

2.3.0.8 Argument description

- `x`: Results from `readGenepop` function
- `y`: Results from `fastDivPart` function. The `WC_Fst` argument in the `fastDivPart` call must be `TRUE`.

2.3.0.9 Example To assess the bias associated with measuring genetic differentiation using F_{ST} type statistics in `Test_data`, the following commands are executed.

```r
# load diveRsity
library(diveRsity)
# load test data
data(Test_data)
Test_data <- Test_data[-33, -33]
# plot the relationship between polymorphism and differentiation
corPlot(Test_data)
```



These results indicate that there is strong bias in these data. The negative trend in the top two plots coupled with the strong positive trend in the bottom two is indicative of loci with high mutation rates. Indeed, one of the major differences between statistics like D_{Jost} and F_{ST} is the former defines differentiation as a process consisting of mutation and genetic drift, while the latter minimizes the role of mutation.

2.4 `diffCalc`

This function allows users to calculate genetic differentiation measures such as F_{ST}, G_{ST}, G'_{ST}, G''_{ST} and D_{Jost}. These parameters can be calculated at various levels including global, per locus and pairwise. Integrated facilities for calculating 95% confidence intervals (CIs) through bootstrapping are also available. The functionality of `diffCalc` overlaps significantly with `fastDivPart`, however the former is implemented more efficiently, making use of significant portion of `C++` code to execute computationally intensive routines. In line with this efficiency, results can only be written to tab delimited files, rather that to xlsx workbooks, as is available in `fastDivPart`.

```
diffCalc(infile = NULL, outfile = NULL, fst = FALSE, pairwise = FALSE,
         bs_locus = FALSE, bs_pairwise = FALSE, boots = NULL, para = FALSE)
```

2.4.0.10 General use

2.4.0.11 Argument description

- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format.
- `outfile`: A character string indicating the prefix to be added to the results directory created. All results files will be written to this directory.
- `fst`: A logical indication of whether Weir & Cockerham's F-statistics should be calculated.
- `pairwise`: A logical argument, indicating if pairwise point estimates should be calculated. Both locus and multilocus parameters are calculated.
- `bs_locus`: A logical argument, indicating whether 95% CIs should be calculated at the global level. If `TRUE` 95% CI are estimated for all loci across all population samples and across all loci for all population samples.
- `bs_pairwise`: A logical argument, indicating whether 95% CIs should be calculated at the pairwise level. If `TRUE` 95% CI are estimated for all loci for all pairwise comparisons and across all loci for all pairwise comparisons.
- `boots`: An integer indicating the number of bootstrap replicates to use for the estimation of 95% CIs.
- `para`: A logical argument specifying whether `diffCalc` should make use of multiple CPUs.

2.4.0.12 Example To calculate pairwise differentiation using D_{Jost} and their statistical significance, the code below can be used:

```
# load diveRsity
library(diveRsity)
# load test data
data(Test_data)
# calculate pairwise stats
res <- diffCalc(infile = Test_data, pairwise = TRUE, bs_pairwise = TRUE,
                boots = 999, para = TRUE)
pander::pandoc.table(head(res$bs_pairwise$D))
```

populations	actual	lower	upper
pop1, vs pop2,	0.0027	-0.0123	0.0195
pop1, vs pop3,	0.1802	0.1514	0.2125
pop1, vs pop4,	0.1484	0.1225	0.1757
pop1, vs pop5,	0.2527	0.223	0.2841
pop1, vs pop6,	0.1494	0.1206	0.1796
pop2, vs pop3,	0.1579	0.1306	0.1879

These results, similar to those from the results from the `chiCalc` example above, indicate that there is a significant differentiation between all population pairs (i.e. lower 95% CI do not overlap zero), except between pop1 and pop2.

2.5 `diffCalc`

This function allows users to visualise pairwise differentiation matrices generated by `fastDivPart` and `diffCalc`. Given that pairwise differentiation estimates can number in the hundreds-thousands, `diffPlot`

allows users to plot values using colour gradients as wells as having interactive tool tips for simple identification of pairwise values.

```
diffPlot(x, outfile = NULL, interactive = FALSE)
```

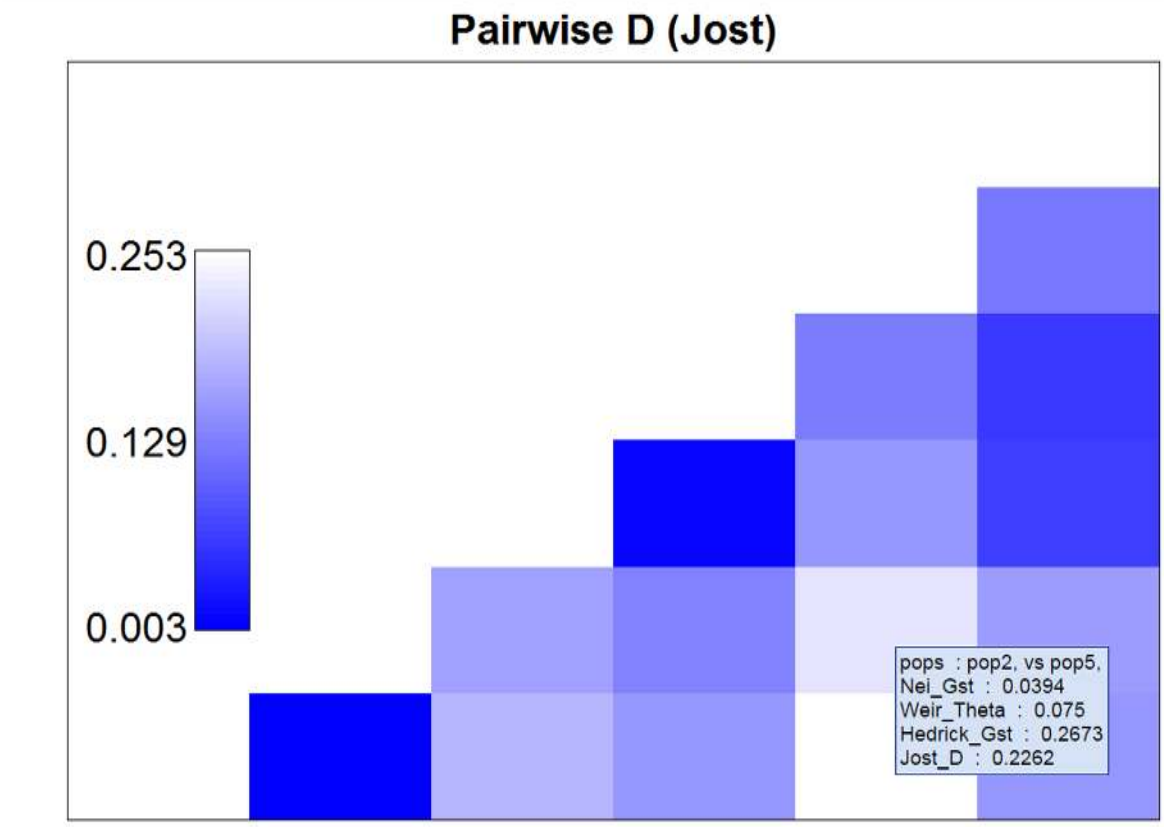
2.5.0.13 General use

2.5.0.14 Argument description

- `x`: Output results from either `fastDivPart` or `diffCalc` where `pairwise = TRUE`.
- `outfile`: A character string indicating the prefix to be added to the results directory created. All results files will be written to this directory.
- `interactive`: A logical indication as to whether the `sendplot` package should be used to generate tool-tip information for plot cells. If `FALSE`, standard .png files will be generated.

2.5.0.15 Example `diffPlot` can be used to easily identify population pairs and their associated genetic differentiation indices.

```
# load diveRsity
library(diveRsity)
# load test data
data(Test_data)
# calculate pairwise stats
x <- diffCalc(Test_data, pairwise = TRUE, fst = TRUE)
diffPlot(x = x, outfile = "Out", interactive = TRUE)
```



As can be seen from this screen cap, hovering the mouse pointer over a cell reveals the associated pairwise comparison as well as the relevant differentiation statistics.

2.6 `divBasic`

This function allows users to calculate a range of descriptive population parameters, including allelic richness, observed and expected heterozygosity, Hardy-Weinberg Equilibrium (HWE) tests and F_{IS}.

Within sample locus HWE tests can be calculated using either chi-square or Fisher's exact goodness of fit testing.

```
divBasic(infile = NULL, outfile = NULL, gp = 3, bootstraps = NULL,
         HWEexact = FALSE, mcRep = 2000)
```

2.6.0.16 General use

2.6.0.17 Argument description

- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format.

- `outfile`: A character string indicating the prefix to be added to the results directory created. All results files will be written to this directory.
- `gp`: An integer indicating the allele digit format of the input genepop file.
- `bootstraps`: An integer indicating the number of bootstrap replicates used to estimate 95% CIs for F_{IS}.
- `HWEexact`: A logical indicating whether Fisher's exact test should be used to test HWE. If `FALSE`, chi-square testing is carried out.
- `mcRep`: An integer indicating the number of Monte Carlo test replicates when using Fisher's exact testing for HWE.

2.6.0.18 Example `divBasic` can be used to calculate a number of useful parameters for population samples. Below is a demonstration of how a simple table can be generated using the function.

```
# load diveRsity
library(diveRsity)
# load test data
data(Test_data)
# calculate pairwise stats
res <- divBasic(infile = Test_data, outfile = NULL, gp = 3, bootstraps = 999,
                HWEexact = TRUE, mcRep = 2000)
pander::pandoc.table(res$mainTab[[1]][,c(1:5, 39)])
```

stats	Locus1	Locus2	Locus3	Locus4	overall
N	46	47	47	47	45.68
A	4	3	11	6	416
%	57.14	100	61.11	66.67	62.04
Ar	3.58	2.93	10.34	5.41	9.89
Ho	0.67	0.57	0.87	0.64	0.68
He	0.66	0.53	0.83	0.67	0.72
HWE	0.539	1	0.997	0.918	0.002
Fis	-0.016	-0.089	-0.049	0.052	0.06
Fis_Low	-0.209	-0.331	-0.152	-0.144	0.029
Fis_High	0.179	0.158	0.062	0.259	0.092

These tables show the descriptive statistics for the first four loci and across all loci for each population sample. A description of each statistic is as follows:

- N: The number of individuals typed per locus. Overall value is the mean number of individuals typed per locus.
- A: The number of alleles observed per locus. Overall value is the total number of alleles observed across all loci.
- %: The percentage of total observed alleles per locus per population sample.
- A_R: Allelic richness per locus. Overall value is the mean allelic richness.
- H_O: Observed heterozygosity per locus. Overall value is the observed heterozygosity across loci.
- H_E: Expected heterozygosity per locus. Overall value is the expected heterozygosity across loci.

- HWE: P-value from a goodness of fit to HWE expectations test using either `chisq.test` (if `HWEexact = FALSE`) or `fisher.test` (if `HWEexact = TRUE`). Overall value is a p-value from either a `chisq.test` where all chi-square differences and degrees of freedom are summed across loci (if `HWEexact = FALSE`), or by combining locus p-values using Fisher's method (if `HWEexact = TRUE`).
- F_{IS}: Wright's inbreeding coefficient per locus. Overall value is the multilocus F_{IS} estimate.
- F_{IS}_Low: The lower limit of the 95% Confidence interval of the corresponding F_{IS} value.
- F_{IS}_High: The upper limit of the 95% Confidence interval of the corresponding F_{IS} value.

2.7 `divMigrate`

This function allows users to explore the patterns of gene flow that have influenced a group of population samples. Accepting a genepop file as input, `divMigrate` allows users to visualise directional components of gene flow and test whether gene flow occurs significantly more in any one direction between pairs of population samples. This can be useful when testing the effects of barriers to gene flow, particularly when gene flow is likely to be impeded in one direction, such as in river systems. Gene flow patters can also be visualized using Network graphs, which can also be saved to file.

```
divMigrate(infile = NULL, outfile = NULL, boots = 0, stat = "all",
           filter_threshold = 0, plot_network = FALSE, plot_col = "darkblue",
           para = FALSE)
```

2.7.0.19 General use

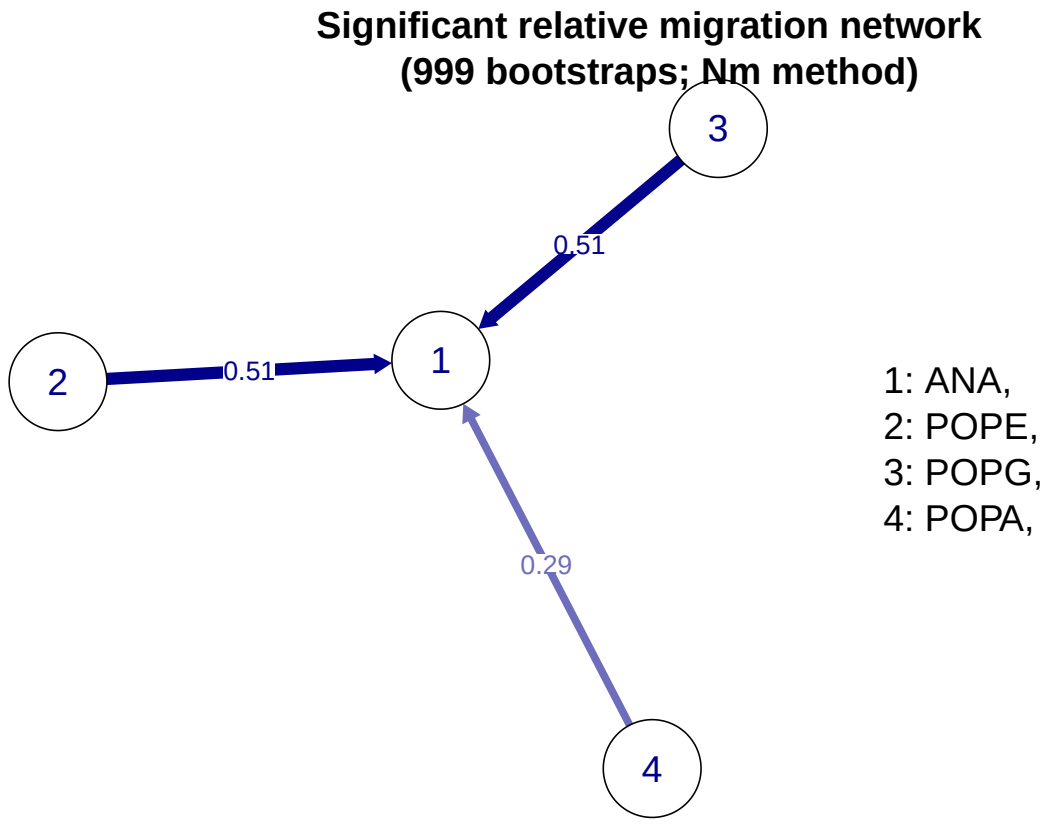
2.7.0.20 Argument description

- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format.
- `outfile`: String prefix to be added to the results directory. If left NULL, no networks will be written to file.
- `boots`: An integer indicating the number of bootstrap replicates to use for the estimation of 95% CIs.
- `stat`: A character string or vector if character strings indicating which statistic should be used to calculate directional gene flow components. Currently three separate methods are available:
  - D_{Jost} (Jost, 2008)
  - G_{ST} (Nei & Chesser, 1983)
  - Nm (Alcala et al, 2014)
- `filter_threshold`: A numeric argument between 0 & 1. This parameter allows users to remove network links less than the value given to reduce noise in network plots. By default, all links between nodes are included in networks, which can result in un-interpretable relationships when many populations are analysed. Increasing the value of `filter_threshold` can help reveal important genetic relationships between groups of populations.
- `plot_network`: A logical argument indicating whether gene flow relationships should be plotted as network graphs.

- `plot_col`: A character string indicating the major colour to be used in the network graph. See the 'Color Specification' section of the `par` help page for more details.
- `para`: A logical argument specifying whether `divMigrate` should make use of multiple CPUs.

2.7.0.21 Example Using a published data set collected from anadromous and river-resident Atlantic salmon (**see chapter x for details**), it is possible to test if gene flow occurs at a significantly higher rate downstream from the river-resident populations to the anadromous population. This hypothesis is based on the presence of multiple barriers to gene flow along the study river. The analysis is shown below.

```
# load diveRsity
library(diveRsity)
# calculate pairwise stats
res <- divMigrate(infile = "s_salar.txt", outfile = NULL, boots = 999,
                  stat = "Nm", plot_network = TRUE, para = TRUE)
```



From the network graph, it is clear that gene flow is occurring at a significantly greater rate from all upstream river-resident populations to the anadromous population.

2.8 `divOnline`

This function allows users to launch a local instance of the `diveRsity-online` web application (online at https://popgen.shinyapps.io/diveRsity-online/). The local version of the application uses local resources, and so is dependent on the systems RAM and CPU.

```
divOnline()
```

2.8.0.22 General usage

2.9 divRatio

Implementing a relatively new method introduced in Skrbinsek et al., (2012), `divRatio` allows users to calculate standardized diversity ratios, relative to a user defined 'yardstick' population sample. The function accepts data two different configurations. + The first, a genepop file containing both the 'yardstick' population sample and all population samples to be compared. If this is the format used, the location of the 'yardstick population should be indicated using the `refPos` argument.

- The second format involves two files. The first file, passed to the `infile` argument should contain the genotype data for the 'yardstick' population sample in the genepop format. The second file, passed to the `pop_stats` argument should be a table in the format shown in the data frame below.

pops	n	alr	alrse	he	hese	validloci
pop2	50	1.701	0.287	0.203	0.084	Loc1 Loc2 Loc3
pop3	50	3.334	0.79	0.428	0.107	Loc1 Loc2 Loc3
pop4	50	2.965	0.412	0.448	0.066	Loc1 Loc2 Loc3
pop5	50	3.357	0.571	0.506	0.089	Loc1 Loc2 Loc3
pop6	50	1.412	0.172	0.106	0.052	Loc1 Loc2 Loc3
pop7	50	4.682	0.461	0.689	0.039	Loc1 Loc2 Loc3

This table contains the following variables:

- `pops`: The name of the populations to be compared to the 'yardstick' population sample
- `n`: Sample size
- `alr`: Mean allelic richness
- `alrse`: The standard error of the mean allelic richness
- `he`: Multilocus expected heterozygosity
- `hese`: The standard error of the multilocus expected heterozygosity
- `validloci`: A character string indicating the names of loci each population sample has in common with the 'yardstick' population sample.

```
divRatio(infile = NULL, outfile = NULL, gp = 3, pop_stats = NULL,
         refPos = NULL, boots = 1000, para = FALSE)
```

2.9.0.23 General use

2.9.0.24 Argument description

- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format. The file can contain either all samples to be analyses, including the 'yardstick' population. Alternatively, if `infile` contains only the 'yardstick'

sample, a table containing the relevant summary statistics for population samples to be compared must be passed to `pop_stats`.

- `outfile`: A character string indicating the prefix to be added to the results directory created. All results files will be written to this directory.
- `gp`: An integer indicating the allele digit format of the input genepop file.
- `pop_stats`: A character string indication the location of a tab delimited file containing the information documented in the table above. A benefit of inputting data in this format is the ability to include population samples scored as variable loci. While all loci for a population sample must also be scored in the 'yardstick' sample, they do not need to be present in the other populations being compared to the 'yardstick' population.
- `refPos`: If `pop_stats = NULL`, this argument specifies the location of the 'yardstick' population sample in `infile`. The argument should be an integer.
- `boots`: An integer specifying the number of bootstrap replicates to use when estimating the standard error for allelic richness and expected heterozygosity per population sample.
- `para`: A logical argument indicating whether the analysis should make use of all available cores on the users system.

2.9.0.25 Example Hypothetically, if the fourth population sample in `Test_data` was suitable for use as a 'yardstick' (see Skrbinsek et al., 2012 for details), the following code outlines the procedure for calculating standardized diversity ratios using `divRatio`.

```
# load diveRsity
library(diveRsity)
# load test data
data(Test_data)
ratios <- divRatio(infile = Test_data, refPos = 4, boots = 999, para = TRUE)
```

pops	n	alr	alrSE	He	HeSE	alrRatio	alrSEratio	heRatio	heSEratio
pop4,-(ref)	53	10.69	1.12	0.74	0.03	1.00	0.15	1.00	0.05
pop1,	47	10.11	1.13	0.72	0.03	0.95	0.15	0.98	0.06
pop2,	42	9.88	1.02	0.73	0.03	0.92	0.14	0.99	0.06
pop3,	41	9.77	1.02	0.74	0.03	0.91	0.14	1.00	0.05
pop5,	41	8.83	0.78	0.74	0.03	0.83	0.11	1.00	0.05
pop6,	41	10.54	1.11	0.76	0.03	0.99	0.15	1.03	0.05

As shown above, `divRatio` produces a table containing both the point estimates of allelic richness (*alr*), and expected heterozygosity (*He*), as well as their respective standard errors (*alrSE* and *HeSE*, respectively) for all population samples. I the table the 'yardstick' population sample is marked with the suffix "-(ref). The variables *alrRatio* and *heRatio* are the allelic richness and heterozygosity, respectively, standardized to the 'yardstick' sample, hence this sample's values are always equal to 1. The variables *alrSEratio* and *heSEratio* are the standard errors of the standardized allelic richness and expected heterozygosity, respectively.

Relative to pop4, all other population samples have slightly lower allelic richness, while all but pop6 have lower expected heterozygosity.

2.10 divSimCo

This function allows the calculation of individual similarity coefficients for all possible pairs of individuals from co-dominant genotype data (ref). Results can be returned for all loci, and across loci. A pairwise distance matrix is returned, which can be used to visualise individual relationships in a phylogenetic tree, using packages such as `ape` or `phanhorn`. An example of this analysis is provided below.

```
divSimCo(infile = NULL, loci = FALSE)
```

2.10.0.26 General use

2.10.0.27 Argument description

- `infile`: A character string specifying the name of the 'genepop' (Rousset, 2008) file from which the statistics are to be calculated This file can be in either the 3 digit of 2 digit format. See http://genepop.curtin.edu.au/help_input.html for detail on the genepop file format..
- `loci`: A logical argument, specifying whether similarity matrices for each locus should be calculated. If FALSE, only a global similarity matrix is given.

2.10.0.28 Example This chapter demonstrates the analyses used in chapter x. The data described in chapter x are used.

The first step in visualising individual relationship is to calculate similarity coefficients among individual using `divSimCo`.

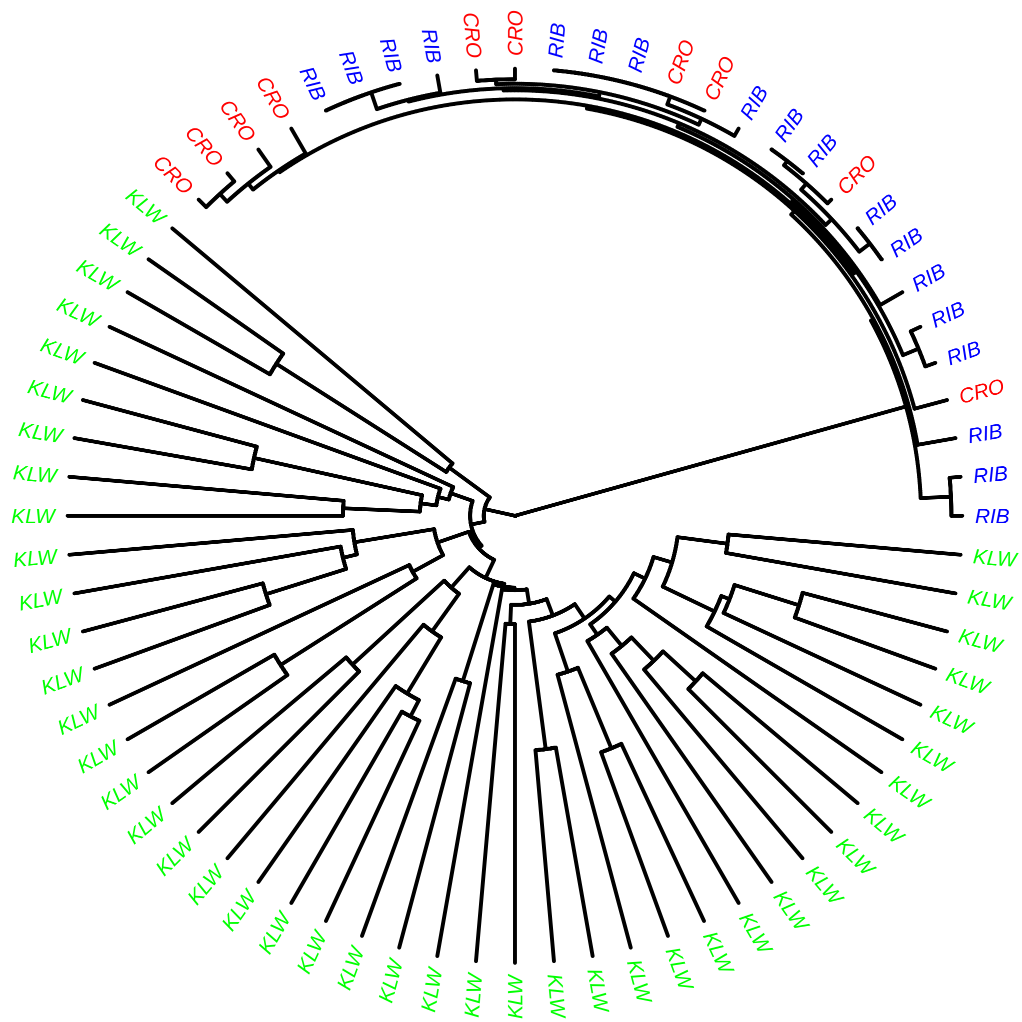
```
# load diveRsity
library(diveRsity)
# calculate pairwise similarity coefficients
dists <- divSimCo(infile = "monomorphism_main.gen")$glb
```

These distance can then be used to draw a UPGMA tree using the `hclust` function to group distances and `plot.phylo` function to draw the tree.

```
library("ape")
tree <- as.phylo(hclust(as.dist(dists), method = "average"))
plot.phylo(tree, type = "u")
```

This tree clearly demonstrates two groups, however, it is difficult to see individuals on it. Below demonstrates a clearer visualisation with coloured tip labels for individuals from different sample sites.

```
# strip names refering to the level to be visualised (e.g. site name)
tree$tip.label <- substr(tree$tip.label, 1, 3)
# create a palette of colours for each unique factor
colLevels <- factor(unique(tree$tip.label))
cols <- rainbow(length(colLevels))
# generate a vector of colours for each tip
tipCols <- sapply(tree$tip.label, function(x){
  cols[which(levels(colLevels) == x)]
})
plot.phylo(tree, label.offset = 0.01, edge.width = 3,
           tip.color = tipCols, type = "f", cex = 1)
```



2.11 `fastDivPart`

Much like the `diffCalc` function, `fastDivPart` provides facilities for users to calculate F_{ST}, G_{ST}, G'_{ST} and D_{Jost} (but not G''_{ST}). All arguments are the same between both functions, however `fastDivPart` requires two additional arguments, `gp` and `plot`. These arguments specify the genepop digit format of `infile` (2-digit or 3-digit), and specifies whether estimated result should be plotted, respectively. The biggest difference between `fastDivPart` and `diffCalc` is speed, `diffCalc` being much more efficient than the former. As such `fastDivPart` provides additional facilities, such as plotting and writing results to multi-sheet excel workbooks. The names of output results are also slightly difference between the two functions.

```
fastDivPart(infile = NULL, outfile = NULL, gp = 3, fst = FALSE,
            pairwise = FALSE, bs_locus = FALSE, bs_pairwise = FALSE,
            boots = NULL, plot = FALSE, para = FALSE)
```

2.11.0.29 General use

2.11.0.30 Argument description

- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format.
- `outfile`: A character string indicating the prefix to be added to the results directory created. All results files will be written to this directory. If the suggested package `xlsx` is installed, all results tables will be written to a single, multi-sheet workbook, alternatively, all tables will be written to separate tab delimited files.
- `gp`: An integer specifying the allele digit format of `infile`. Either '2' or '3' are valid.
- `fst`: A logical indication of whether Weir & Cockerham's F-statistics should be calculated.
- `pairwise`: A logical argument, indicating if pairwise point estimates should be calculated. Both locus and multilocus parameters are calculated.
- `bs_locus`: A logical argument, indicating whether 95% CIs should be calculated at the global level. If `TRUE` 95% CI are estimated for all loci across all population samples and across all loci for all population samples.
- `bs_pairwise`: A logical argument, indicating whether 95% CIs should be calculated at the pairwise level. If `TRUE` 95% CI are estimated for all loci for all pairwise comparisons and across all loci for all pairwise comparisons.
- `boots`: An integer indicating the number of bootstrap replicates to use for the estimation of 95% CIs.
- `plot`: A logical argument allowing users to control whether estimated results should be plotted. Plots are either generated using the standard png device, or if the suggested package `sendplot` is installed, interactive HTML format plots will be created, allowing users to explore elements using the associated tootip information generated.
- `para`: A logical argument specifying whether `fastDivPart` should make use of multiple CPUs.

2.11.0.31 Example To calculate pairwise differences using D_{Jost} and their statistical significance, the code below can be used:

```
# load diveRsity
library(diveRsity)
# load test data
data(Test_data)
# calculate pairwise stats
res <- fastDivPart(infile = Test_data, pairwise = TRUE, bs_pairwise = TRUE,
                   boots = 999, para = TRUE)
pander::pandoc.table(head(res$bs_pairwise$djostEst))
```

Table 5: Table continues below (continued below)

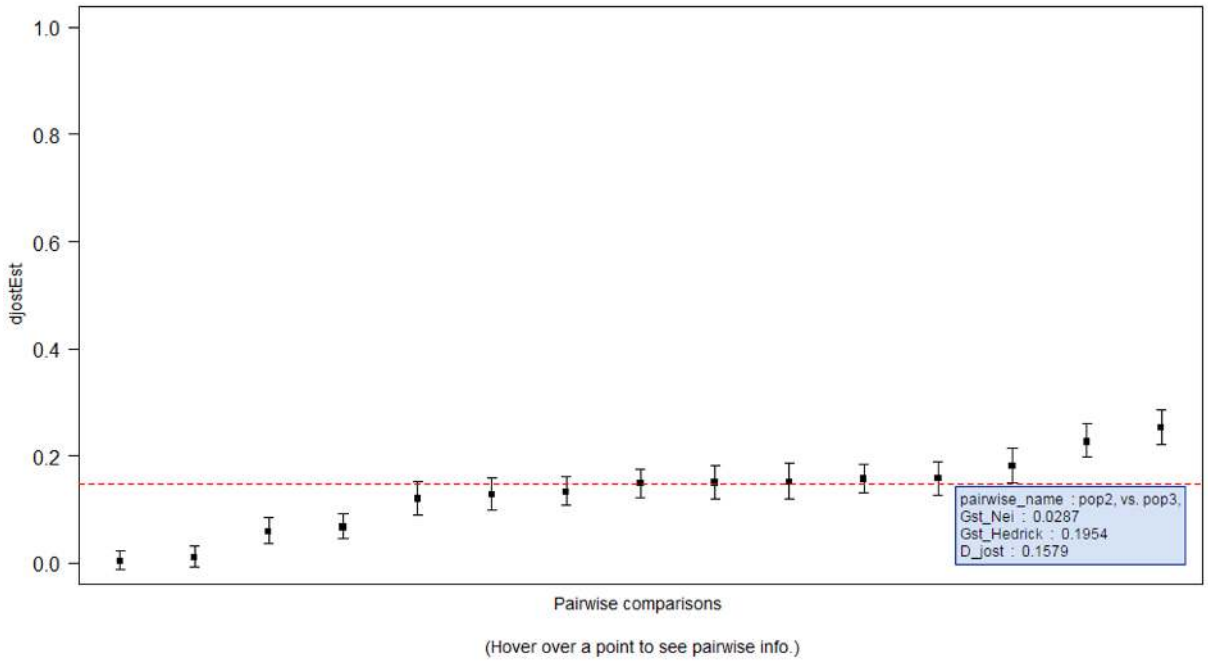
	actual	mean	BC_mean	Lower_95%CI
pop1, vs. pop2,	0.002728	0.0317	0.002728	0.01702
pop1, vs. pop3,	0.1802	0.2094	0.1802	0.1782
pop1, vs. pop4,	0.1484	0.1762	0.1484	0.1506
pop1, vs. pop5,	0.2527	0.274	0.2527	0.2423
pop1, vs. pop6,	0.1494	0.1847	0.1494	0.1565
pop2, vs. pop3,	0.1579	0.1883	0.1579	0.1605

	Upper_95%CI	BC_Lower_95%CI
pop1, vs. pop2,	0.05095	-0.01195
pop1, vs. pop3,	0.2438	0.149
pop1, vs. pop4,	0.2038	0.1227
pop1, vs. pop5,	0.3082	0.221
pop1, vs. pop6,	0.2149	0.1212
pop2, vs. pop3,	0.2207	0.1301

	BC_Upper_95%CI
pop1, vs. pop2,	0.02198
pop1, vs. pop3,	0.2146
pop1, vs. pop4,	0.1759
pop1, vs. pop5,	0.2869
pop1, vs. pop6,	0.1796
pop2, vs. pop3,	0.1903

As in differentiation results from both `diffCalc` and `chiCalc`, only the difference between pop1 and pop2 is non-significant. The pairwise bootstrapping results from `fastDivPart` contains both standard 95% CIs (*Lower_95%CI* and *Upper_95%CI*) and bias corrected 95% CIs (*BC_Lower_95%CI* and *BC_Upper_95%CI*), while `diffCalc` only returns the bias corrected CI.

If the argument `plot = TRUE` was used, and the suggested package `sendplot` was installed, a plot with tool-tip information, as below, can be generated. An argument must be passed to `outfile` to allow the function to write the plots to file, otherwise standard png plots will be returned to the plot device.



2.12 gpSampler

The ability to randomly sub-sample population data 'in silico' is essential for investigations such as sample effects on parameter estimation. By integrating this process into the `R` workflow with `gpSampler`, the `diveRsity` package allows users to conveniently generate sub-sampled data and estimated various parameters from them.

```
gpSampler(infile = NULL, samp_size = 10, outfile = NULL)
```

2.12.0.32 General use

2.12.0.33 Argument description

- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format.
- `samp_size`: An integer vectors specifying the size of sub-sample to be taken from each population sample in `infile`. If a single integer is provided, all population samples in `infile` will be re-sampled for the same number of individuals. Alternatively, each sample in `infile` will be sub-sampled for a number of individuals specified by the corresponding value in `samp_size`. For example, if `samp_size = c(10, 5, 10)`, then the first and third population sample in `infile` will be sub-sampled for n = 10, while the second population sample will be sub-sampled for n = 5.
- `outfile`: A character string indicating the prefix to be added to the .gen file containing the sub-sampled data.

2.12.0.34 Example This example demonstrates how `gpSampler` can be used to understand the effect of sample size on the calculation of D_{Jost}. The main challenge from this task is the organisation of data. This example provides a suitable strategy for such a task.

Here, each population sample will be re-sampled for $n = 10, 20, 30, 40, 1000$ times and D_{Jost} will be estimated for each sub-sample of the data. These results will then be plotted with a +/- 1 standard deviation envelope, calculated from the sub-sample data

```
# load diveRsity
library(diveRsity)
# generate sub-sample ouput names
samp_sizes <- seq(10, 40, 10)
outnms <- lapply(samp_sizes, function(x){
  # generate the sample size folder
  dir.create(path = paste("./n", x, sep = ""))
  # create oufile names
  lapply(1:1000, function(i){
    c(x, paste("./n", x, "/", x, "_rep_", i, ".gen", sep = ""))
  })
})
# generate subsample data
sapply(outnms, function(x){
  sapply(x, function(y){
    gpSampler(infile = "Test_data.txt", samp_size = as.numeric(y[[1]]),
              outfile = y[[2]])
    })
})
```

When all of the sub-sample files are generated, it is straightforward to pass the file names to `diffCalc` to calculate overall D_{Jost}. This is done as follows:

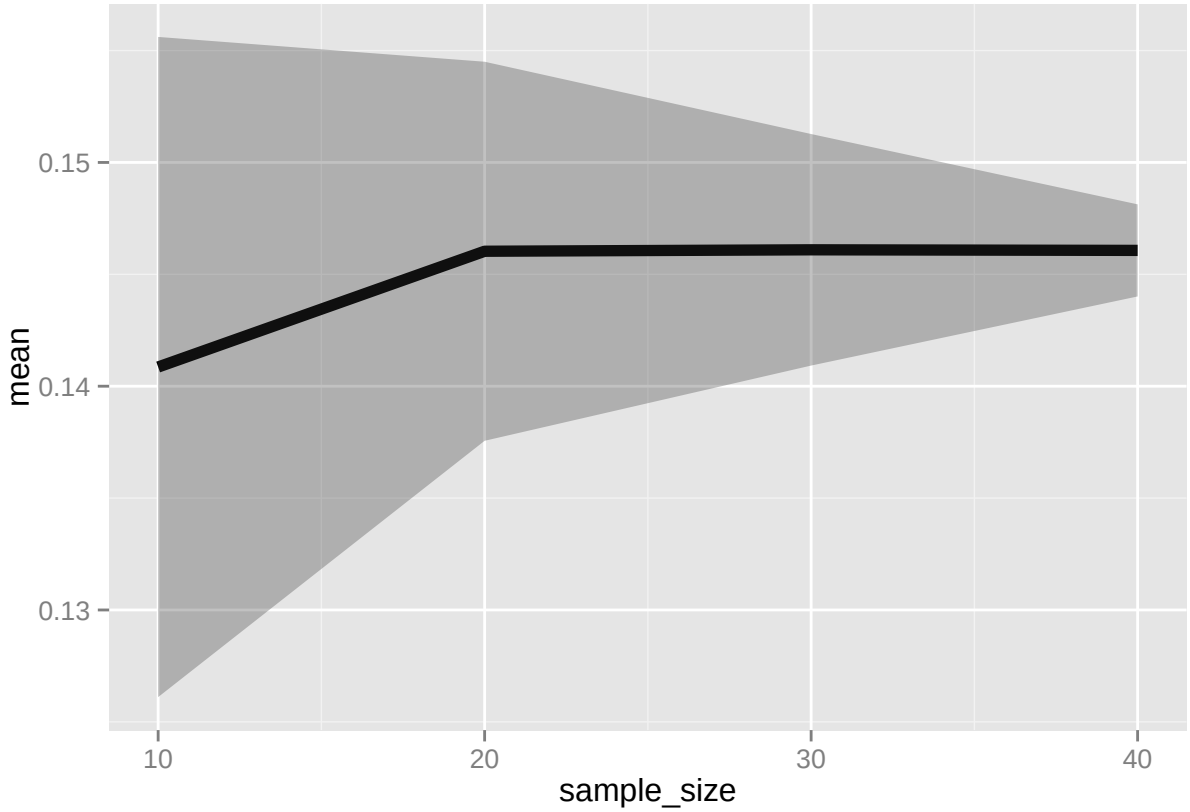
```
# calculate Jost's D
d <- lapply(outnms, function(x){
  sapply(x, function(y){
    diffCalc(y[[2]])$std_stats[38,"D"]
  })
})
```

Estimating both the mean and the standard deviation for each of the four samples sizes tested is done as follows:

```
plot_stats <- t(sapply(d, function(x){
  mn <- mean(x, na.rm = TRUE)
  std <- sd(x, na.rm = TRUE)
  c(mn, mn-std, mn+std)
}))
plot_stats <- as.data.frame(plot_stats)
plot_stats$sample_size <- samp_sizes
names(plot_stats) <- c("mean", "lower", "upper", "sample_size")
```

To visualize the effect of sample size on the estimation of D_{Jost}, these results can be plotted as follows:

```r
# load ggplot2
library(ggplot2)
# create the plot
plt <- ggplot(data = plot_stats, aes(x = sample_size, y = mean)) +
  geom_line(lwd = 2) +
  geom_ribbon(aes(ymin=lower, ymax=upper), alpha = 0.3)
plt
```



From these results, it can be seen that the variance in D_{Jost} is much higher at smaller sample sizes, decreasing as sample size increases. Additionally, the results indicate a slight underestimation of D for small sample sizes.

2.13 inCalc

Locus informativeness for the inference of ancestry (I_n) can be calculated using `inCalc`. The function accepts an input file in the genepop format, and allows users to calculate this parameter, both globally and for pairs of populations.

```r
inCalc(infile = NULL, outfile = NULL, pairwise = FALSE, xlsx = FALSE,
       boots = NULL, para = FALSE)
```

2.13.0.35 General use

2.13.0.36 Argument description

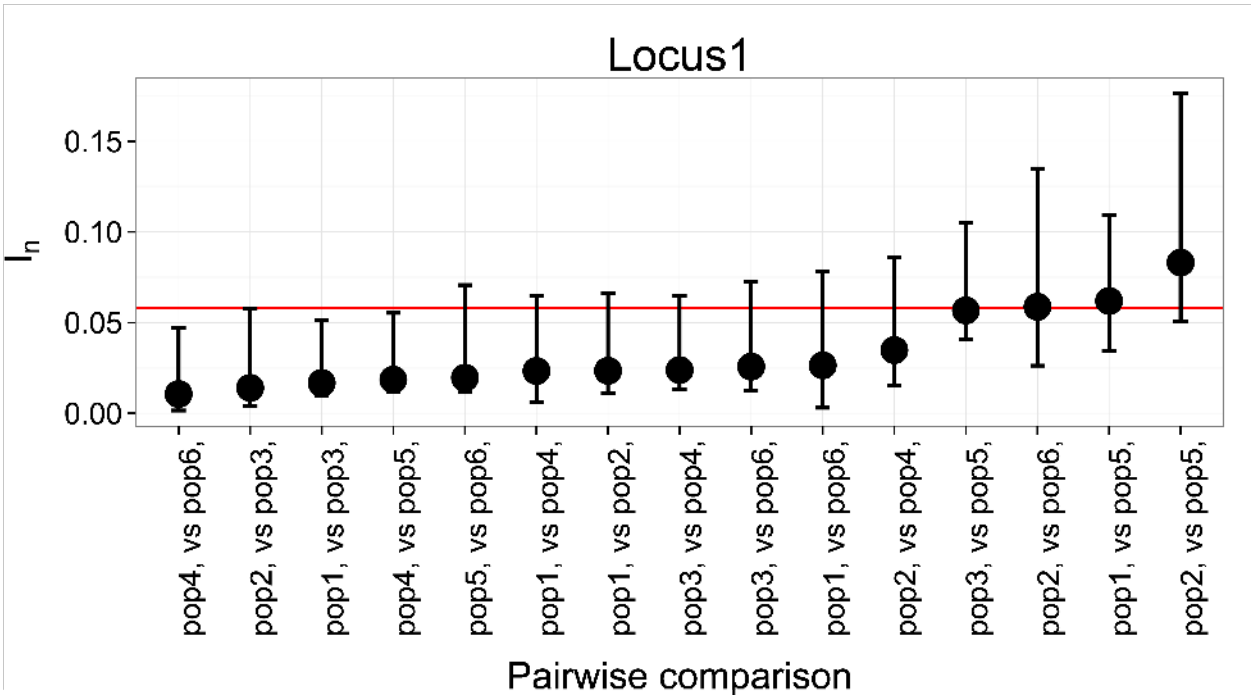
- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format.
- `outfile`: A character string indicating the prefix to be added to the results directory created. All results files will be written to this directory.
- `pairwise`: A logical argument specifying whether I_n should be calculated for all population pairs.
- `xlsx`: A logical argument indicating whether users would like results to be written to a multi-sheet xlsx workbook. If false, result will be written to tab delimited files.
- `boots`: An integer specifying the number of bootstrap replicates used when calculating 95% confidence intervals.
- `para`: A logical argument indicating whether the function should make use of multiple cores.

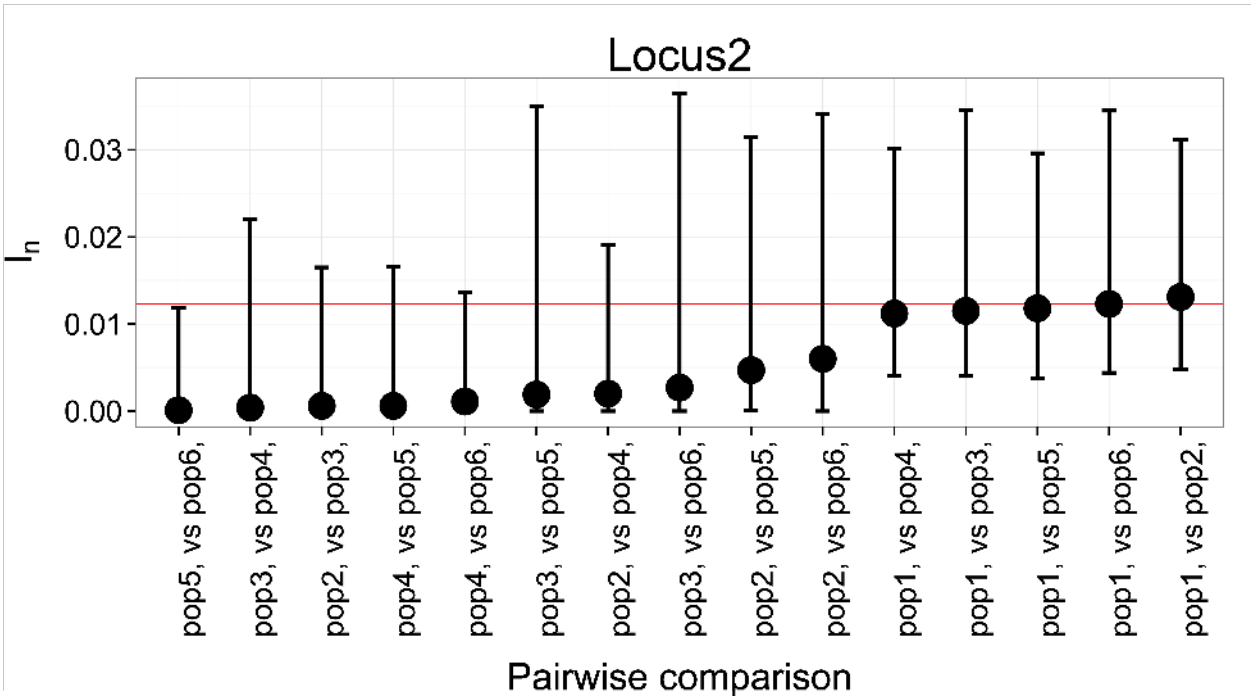
2.13.0.37 Example When selecting a panel of markers for studying genetic structure among populations, it is important that these loci are generally informative across multiple populations. For instance, a number of factors can affect the informativeness of a locus, such that they may be informative for one pair of populations, but not another. This example demonstrates how informativeness can be calculated and visualized with the help of the `inCalc` function.

Calculate pairwise I_n with 95% CIs

```
# load diveRsity
library("diveRsity")
data(Test_data)
in_res <- inCalc(Test_data, pairwise = TRUE, boots = 999, para = TRUE)
```

```
# load ggplot2
library("ggplot2")
loci <- as.character(in_res$global$Locus)
for(i in 1:(nrow(in_res$pairwise)-1)){
  loc1 <- data.frame(t(in_res$pairwise[i,-1]),
                     t(in_res$lower_CI[i,-1]),
                     t(in_res$upper_CI[i,-1]))
  colnames(loc1) <- c("actual", "lower", "upper")
  loc1$pops <- rownames(loc1)
  rownames(loc1) <- NULL
  p <- ggplot(data = loc1, aes(x = reorder(pops, actual), y = actual)) +
    geom_hline(yintercept = in_res$global$Global_In[i], colour = "red") +
    geom_point(pch = 20, cex = 10) +
    geom_errorbar(aes(ymin = lower, ymax = upper),
                  width = 0.2, size = 1) +
    theme_bw() +
    theme(text = element_text(size=20),
          axis.text.x = element_text(angle=90, vjust=1)) +
    ggtitle(loci[i]) +
    labs(x = "Pairwise comparison", y = expression("I"["n"]))
  ggsave(paste("In_Locus", i, ".eps", sep = ""), device = cairo_ps)
}
```





From these plots we are able to visualise the variance in I_N among pairwise comparisons. Each plot represent pairwise I_n calculates for a given locus along with 95% CIs. The red line in each plot is the global I_n. This approach can be used to identify loci with have high informativeness for the inference of ancestry at multiple scales. Genepop input files can be organised to represent different hierarchical scales, allowing a more detailed inspection of the informativeness of loci for specific question of interest.

2.14 microPlexer

This function allows users to launch a local instance of the microPlexer web application (http://glimmer.rstudio.com/kkeenan/microPlexer/). The local version of the application uses local resources, and so is dependent on the systems RAM and CPU.

This web application allows users to generate efficient configurations of microsatellite loci for the purposes of multiplex PCR. The configurations generated only take into consideration the number of dye channels available on the screening platform to be used, and the specified size range of the loci. It does not assess the annealing temperature of primers etc.

```
microPlexer()
```

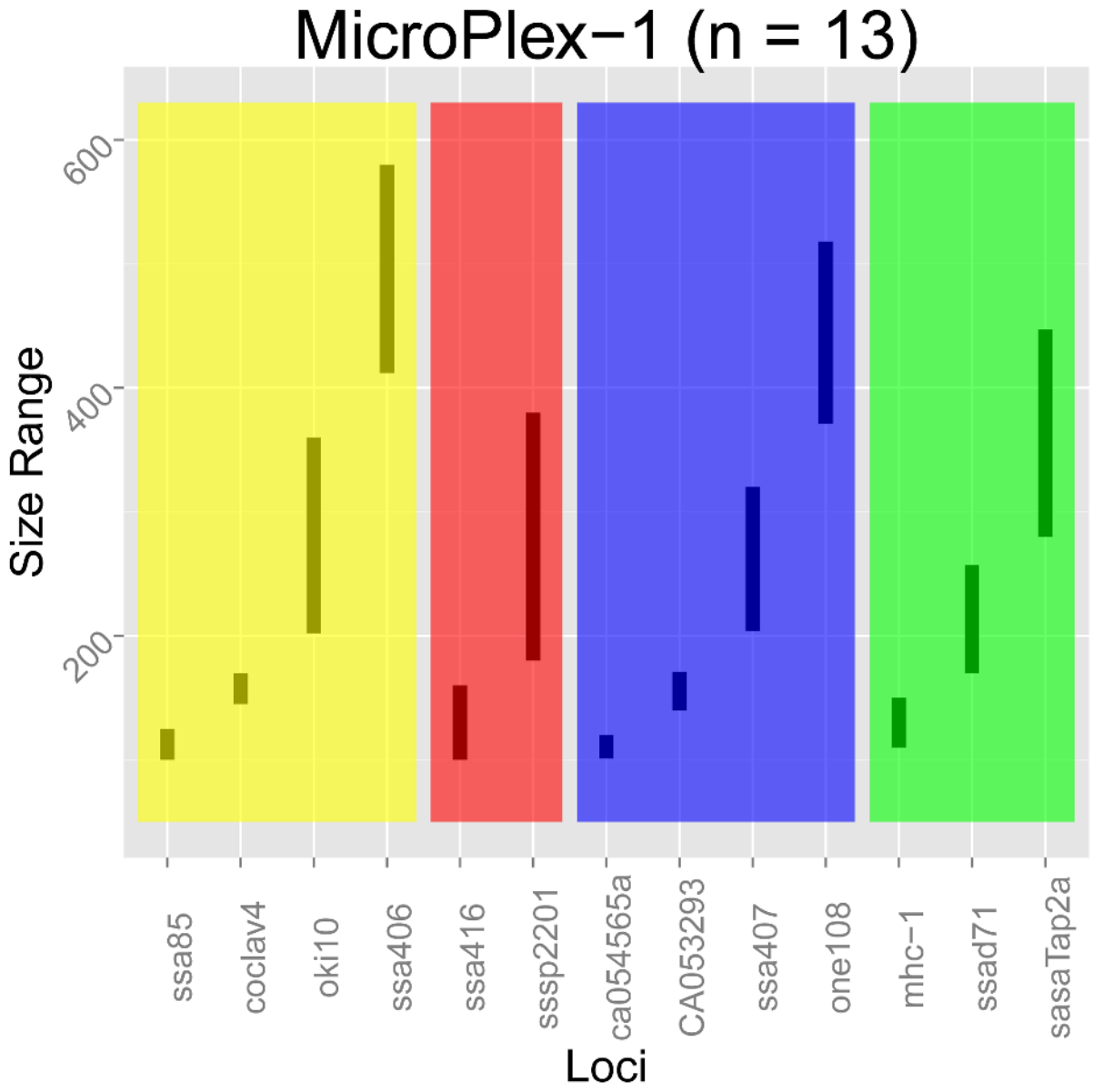
2.14.0.38 General usage

2.14.0.39 Example The application accepts a .csv file as input with the following structure:

nms	lower	upper
ssa85	100	125
one102b	160	280
ssa406	412	580
mhc-1	110	150
ca048302	172	219
ssa419	230	570

Users are able to specify the number of dyes available for screening and the minimum distance between loci within dye groups. Additionally, two algorithms are available for use when grouping loci. These are 'Maximum throughput' and 'Balanced throughput'. Maximum throughput attempts to group loci into as few a number of multiplexes as possible, meaning that, depending on the total number of loci and their size ranges, the first multiplex will generally have the largest number of loci. By using the 'Balanced throughput' option, user can specify a mean number of loci per multiplex group. In this case the application will attempts to balance the number of loci in each multiplex.

Following plot represent a typical, single multiplex groups generated by microPlexer:



2.15 `polyIn`

This function is virtually identical to `inCalc` in that it calculates locus informativeness for the inference of ancestry at global and pairwise scales. However, `polyIn` allows the use of loci with arbitrary ploidy. Results are also only returned to the R workspace, so users must write them to file if required. This function does not calculate 95% CIs.

The input file accepted is a modified genepop format. An example of this file, containing tetraploid is below:

```
pop-test
snp1    snp2    snp3    snp4    snp5
pop
pop1_1  AABA    AABA    AABA    AABA    AABA
```

```
pop1_2  AAAA    AAAA    AAAA    AAAA    AAAA
pop1_3  AABB    AABB    AABB    AABB    AABB
pop1_4  BBBB    BBBB    BBBB    BBBB    BBBB
pop1_5  AABA    AABA    AABA    AABA    AABA
pop1_6  ABAB    ABAB    ABAB    ABAB    ABAB
pop1_7  BBAA    BBAA    BBAA    BBAA    BBAA
pop1_8  AABB    AABB    AABB    AABB    AABB
pop1_9  ABAA    ABAA    ABAA    -9      ABAA
pop1_10 AAAA    AAAA    AAAA    AAAA    AAAA
pop
pop2_1  AABB    AABB    AABB    AABB    AABB
pop2_2  AABB    AABB    AABB    AABB    AABB
pop2_3  BBBB    BBBB    BBBB    BBBB    BBBB
pop2_4  BBBA    BBBA    BBBA    BBBA    BBBA
pop2_5  BBBA    BBBA    BBBA    BBBA    BBBA
pop2_6  BABB    BABB    BABB    BABB    BABB
pop2_7  ABBB    ABBB    ABBB    ABBB    ABBB
pop2_8  AAAB    AAAB    AAAB    AAAB    AAAB
pop2_9  ABAB    ABAB    ABAB    ABAB    ABAB
pop2_10 BBAA    BBAA    BBAA    BBAA    BBAA
pop
pop2_1  AABB    AABB    AABB    AABB    AABB
pop2_2  AABB    AABB    AABB    AABB    AABB
pop2_3  BBBB    BBBB    BBBB    BBBB    BBBB
pop2_4  BBBA    BBBA    BBBA    BBBA    BBBA
pop2_5  BBBA    BBBA    BBBA    BBBA    BBBA
pop2_6  BABB    BABB    BABB    BABB    BABB
pop2_7  ABBB    ABBB    ABBB    ABBB    ABBB
pop2_8  AAAB    AAAB    AAAB    AAAB    AAAB
pop2_9  ABAB    ABAB    ABAB    ABAB    ABAB
pop2_10 BBAA    BBAA    BBAA    BBAA    BBAA
```

As can be seen, the format of the file follows that of the standard genepop format. However, missing data is recorded as '-9' and genotypes are recoded as shown, rather than as allele sizes/numbers.

2.15.0.40 Argument description

- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format.
- `pairwise`: A logical argument specifying whether I_n should be calculated for all population pairs.
- `parallel`: A logical argument indicating whether the function should make use of multiple cores.

2.15.0.41 Example Using the file above, it is possible to calculate global I_n for all loci as follows:

```
library(diveRsity)
polyIn("polyIn_data.gen")
```

```
      snp1       snp2       snp3       snp4       snp5
0.02810769 0.02810769 0.02810769 0.02506269 0.02810769
```

Here, all five loci appear to be equally informative for the inference of ancestry, at the level explored.

2.16 readGenepop

This function allows the generation of various objects from 3 or 2 digit genepop files. These objects can be used directly (e.g. allele frequency plots below), or to efficiently construct user defined functions.

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

2.16.0.42 General use

2.16.0.43 Argument description

- `infile`: A character string indicating the location and name of a genepop format file to be read. If the file is in the current working directory, only the name must be provided. If the file is in a directory other than the current working directory, either a relative or absolute path to the file must be provided. The genepop file can be in the 2-digit or 3-digit allele format.
- `gp`: An integer specifying the digit format of the input genepop file (e.g. 2-digit or 3-digit).
- `bootstrap`: A logical argument allowing users to generate a genepop file object containing re-sampled data. This utility is designed to allow users to generate re-sampled data sets from which 95% CIs for a given parameter can be estimated.

2.16.0.44 Example Sometimes just visualising the differences in allele frequencies at a locus among population samples can be informative. Here, `readGenepop` can be used to generate allele frequency matrices, which can then be manipulated to generate stacked bar chart of these data.

```
library("diveRsity")
data("Test_data")
res <- readGenepop(Test_data)
af <- res$allele_freq
```

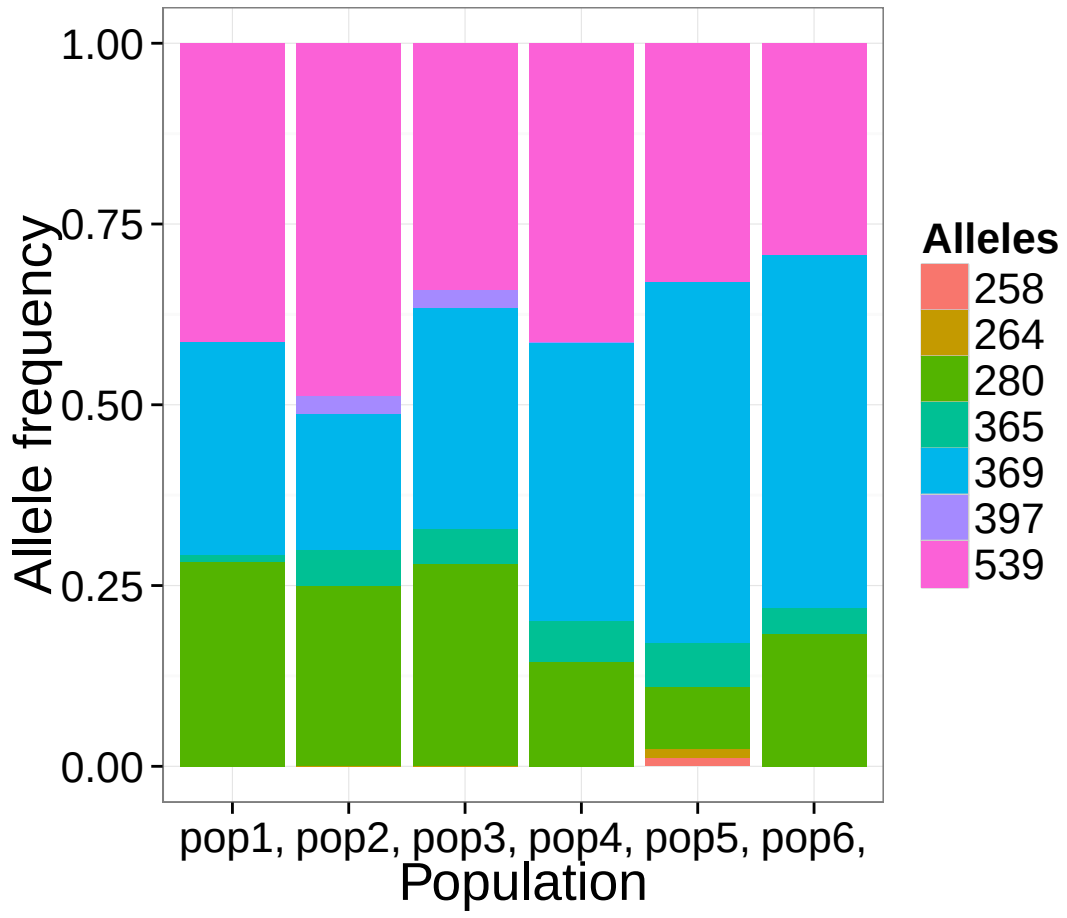
These commands result in a list of matrices (`af`). Each element of the list has the following structure:

```
af[[1]]
```

```
          [,1]   [,2]       [,3]       [,4]       [,5]       [,6]
258 0.00000000 0.0000 0.00000000 0.00000000 0.01219512 0.00000000
264 0.00000000 0.0000 0.00000000 0.00000000 0.01219512 0.00000000
280 0.28260870 0.2500 0.28048780 0.14423077 0.08536585 0.18292683
365 0.01086957 0.0500 0.04878049 0.05769231 0.06097561 0.03658537
369 0.29347826 0.1875 0.30487805 0.38461538 0.50000000 0.48780488
397 0.00000000 0.0250 0.02439024 0.00000000 0.00000000 0.00000000
539 0.41304348 0.4875 0.34146341 0.41346154 0.32926829 0.29268293
```

To plot these data, the following manipulations can be done:

```
library("reshape2")
library("ggplot2")
colnames(af[[1]]) <- res$pop_names
loc1 <- melt(af[[1]])
p <- ggplot(data = loc1, aes(x = Var2, y = value, fill = factor(Var1))) +
  geom_bar(stat = "identity") +
  theme_bw() +
    theme(text = element_text(size=20)) +
    labs(x = "Population", y = "Allele frequency", fill = "Alleles")
print(p)
```



2.17 snp2gen

This function allows users to efficiently convert SNP genotypes into the genepop format for further use with the `diveRsity` package. The function accepts a matrix stored in a tab delimited file as below:

```
SNP_ID  pop1_1  pop1_2  pop1_3  pop1_4  pop1_5
SNP1    TC  TC  TC  TC  TC
SNP2    TC  TC  TC  TC  TC
SNP3    TA  TA  TA  TA  AA
SNP4    AG  AG  AG  AG  AG
```

```
SNP5    TC  TC  TC  TC  TC
```

This file contains genotype data for five individuals at five SNP loci. Individuals are stored in columns, and loci in rows.

```
snp2gen(infile = NULL, prefix_length = 2, write = FALSE)
```

2.17.0.45 General use

2.17.0.46 Argument description

- `infile`: A character string pointing to an SNP matrix file to be converted. This string can either be a file name, providing the file is in the current working directory, otherwise an absolute or relative file path can be provided within the string.
- `prefix_length`: An integer indicating the number of characters from the beginning of the individual ID strings to be used to group individuals into population samples.
- `write`: A logical argument specifying whether a genepop file containing the converted data should be written to disk. If `FALSE`, the genepop file will be returned to the R workspace. This can be passed to other functions from the `diveRsity` package. Thus avoiding the time cost of reading the file twice, especially useful when converting large data sets (i.e. $>$ 10,000 loci).

2.17.0.47 Example Below is the output for the example input file above:

```
snp2gen-converted
SNP1,   SNP2,   SNP3,   SNP4,   SNP5
pop
pop1_1 ,    0402    0402    0401    0103    0402
pop1_2 ,    0402    0402    0401    0103    0402
pop1_3 ,    0402    0402    0401    0103    0402
pop1_4 ,    0402    0402    0401    0103    0402
pop1_5 ,    0402    0402    0101    0103    0402
```

3 References

in progress