diveRsity v1.6.1 Help Manual

by *Kevin Keenan*

kkeenan02@qub.ac.uk

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

December 7, 2016

Contents

1 Introduction

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

1.1 About R

`R` is an extremely powerful and popular software for statistical programming. It is very well supported by a dedicated group of people known as the *R core development team* (R Development Core Team, 2011a), as well as an active community of developers and use`R`s. More information about R can be found at `http://www.r-project.org/about.html`.

1.2 About diveRsity

`diveRsity` is a package containing multiple 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, observed/expected heterozygosity and Hardy-Weinberg tests. The package also provides functions for the calculation of Weir & Cockerham's 1984 F-statistics and 'Yardstick' diversity ratios from Skrbinšek *et al.,* (2012).

In addition to these features, `diveRsity` also provides users with various options 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.

`diveRsity` was written to ensure that even `R` beginners can carry out genetic analyses in `R` without major difficulties. By automatically writing analysis results to file, use`R`s do not need to understand how to access `variables` in the `R` environment, let alone know what a `variable` is. However, for more experienced use`R`s, all analysis functions return results `variables` to the `R` environment, details of which are provided in the **Function usage** section below.

1.2.1 How to cite

Keenan, K., McGinnity, P., Cross, T. F., Crozier, W. W., & Prodöhl, P. A. (2013). `diveRsity`: An R package for the estimation and exploration of population genetics parameters and their associated errors. *Methods in Ecology and Evolution*, 4: 782-788. doi:10.1111/2041-210X.12067

1.2.2 What's new?

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

Parallel computations are also now available when using the `inCalc` and `divPart` 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 `divPart`. This additional functionality now allows useRs to calculate pairwise statistics without having to run the computationally intensive bootstrap algorithm, thus saving time.

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

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 execution time if a large number of bootstrap iterations are used. For this reason, the calculation of F-statistics has been included as an optional extra through the new argument `WC_Fst`.

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

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

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

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

As of version 1.3.6, a web app version of `diveRsity` is packaged with the `R` console version. This application can be launched simply by typing:

```
> divOnline()
```

An online version of the app is also available at:
http://glimmer.rstudio.com/kkeenan/diveRsity-online/

This web app allows users to carry out most of the analyses provided by the `divPart` function, with additional plotting options.
Version 1.3.6 also contains a new function allowing the calculation of genetic heterogeneity, using X^2 tests. This new function is named `chiCalc`.

Version 1.4.2 introduces a new function, `divBasic`. This function calculates multiple population sample specific parameters including; allelic richness, observed & expected heterozygosity and Hardy-Weinberg equilibrium χ^2 tests. See below for more details on its' usage.

Version 1.4.4 represents a major update of `diveRsity`, introducing two new functions, `fstOnly` and `divRatio`, as well as the deprecation of some old function (`div.part`, `in.calc` & `readGenepop.user`). `fstOnly` calculates only Weir and Cockerham's 1984, θ and F for loci, global and pairwise levels. Bootstrapped confidence intervals can also be calculated. The function is intended for use by those with very large data sets as it should be more memory efficient than `divPart` which also calculates these statistics along with a variety of other parameters. The function `divRatio` calculates the allelic richness and expected heterozygosity standardised ratios originally presented

by (Skrbinšek *et al.*, 2012).

Version 1.4.6 is the first in a series of package releases with developments focusing on the analysis of large SNP data sets. This release contains a new function, `bigDivPart`, which allows users to calculates the same parameters as `divPart` for large data sets containing thousands of marker loci. The function only accepts genepop format files currently, but work is ongoing to allow for additional formats. No bootstrapping procedures are yet available for this function. These are also under development. In any case, given the massive computational effort involved in bootstrapping large data sets (e.g. 500 individuals across 10 population samples, genotyped for 100,000 SNP loci), these procedures may have to be restricted to High Performance Computing (HPC) environments.

Version 1.5.0 now contains citation information for the `diveRsity` package (Keenan *et al.*, 2013). Users can extract `bibtex` information for `diveRsity` by typing the following into the `R` console:

```
> citation("diveRsity")
```

Version 1.5.0 also fixes a major bug on Mac systems. Due to unexpected behaviour of the `sprintf` function, `divPart`, `inCalc` and `readGenepop` were incorrectly coding alleles, thus leading to erroneous parameter calculations.

Version 1.5.3 introduces the new web app, `microPlexer`. This application is intended to aid researchers wishing to develop microsatellite multiplex system based on locus size ranges and fluorphore tags.

Version 1.5.6 introduces a new function `arp2gen`, which allows users to convert *Arlequin* genotype files to *genepop* files. The function is intended to provide a more integrated experience to users by negating the need to use separate software to carry out file conversions. Although this function

only allows **.arp** to **.gen** file conversions, other `R` packages provide additional conversion facilities. Some such packages are; `adegenet` and `PopGenKit`.

Version 1.5.7 introduced a new **experimental** function, `divMigrate`. The function is based on a new methods presented in (Sundqvist *et al.*, 2013), for detecting the direction of differentiation, and using this information to infer the relative migration between pairs of populations. While the method performs well for some simple migration scenarios, it is still not clear how generally applicable it is under other scenarios (e.g. the effect of high mutation rates, or recent divergence etc.). For this reason, we recommend caution when applying the method to your own data.

Version 1.6.0 contains a new function, `fastDivPart`. This function is virtually identical to `divPart`, with some slight changes to output structures and formats. The key differences in these formats are:

1. `$pairwise` in `fastDivPart` only contains the unbiased estimators of G_{ST}, G'_{ST}, D_{Jost} and θ.

2. The names of the output data structures in `$pairwise`, `$bs_pairwise` and `$meanPairwise` are now `gstEst`, `gstEstHed`, `djostEst` and `thetaWC`, rather than `Gst_est`, `G_hed_st_est`, `D_Jost_est` and `Fst_WC`, respectively.

This new function is optimised to provide faster pairwise calculations, especially where the number of pairwise comparisons is much greater than the number of loci. Some performance tests can be found at:
https://github.com/kkeenan02/diveRsity-dev/blob/master/fastDivPart-tests.md

IMPORTANT NOTE
Users are encouraged to use `fastDivCalc` rather than `divPart`, as the latter function will eventually be removed from future releases of `diveRsity`.

Version 1.6.1 contains a new function, `haploDiv`, for calculating Weir &

Cockerham's θ from haploid genotype data. The data should be in the two or three digit genepop format. See below for further description of the function.

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

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

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

The package is updated regularly, both with added functionality and bug
fixes. The most up to version of `diveRsity` can be installed from `github`
using the `devtools` package:

```
> # install and load devtools
> install.packages("devtools")
> library("devtools")
> # download and install diveRsity
>
> install_github(name = "diveRsity", subdir = "kkeenan02")
```

2.3 Installing optional enhancer packages

The dependencies `plotrix` (Lemon, 2006) and `shiny` (RStudio & Inc., 2012) will download automatically if you install `diveRsity` from `CRAN`. Suggested/optional packages should be installed manually (excluding `parallel`, which is distributed with `R`).

Optional packages are:

`xlsx` — writes results to .xlsx. (Dragulescu, 2012)

`sendplot` — Plots results to .html files with tool-tip information. (Gaile *et al.*, 2012)

`doParallel` — Used in parallel computations. (Revolution Analytics, 2012a)

`parallel` — Used in parallel computations. (R Development Core Team, 2011b)

`foreach` — Used in parallel computations. (Revolution Analytics, 2012b)

`iterators` — Used in parallel computations. (Revolution Analytics, 2012c)

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

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

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

2.4 Loading diveRsity

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

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

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

For convenient access to usage information on each function, within the `R` environment, type:

```
> ?divPart
> ?inCalc
> ?readGenepop
> ?corPlot
> ?difPlot
> ?chiCalc
> ?divOnline
> ?divBasic
> ?fstOnly
> ?divRatio
> ?microPlexer
> ?arp2gen
> ?divMigrate
> ?haploDiv
> ?fastDivPart
```

Each of these commands will provide information on function usage. The

help pages associated with each function describe in detail how each argument should be passed to the function.

3 Function details

3.1 divPart()

NOTE

This function was previously known as `div.part`. This name has since been deprecated. Please use `divPart` instead. `divPart` (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} **(Nei, 1973; Nei & Chesser, 1983)**

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

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

G'_{ST} **(Hedrick, 2005)**

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

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

D_{Jost} **(Jost, 2008)**

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

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

3.1.2 Estimator formulae

The estimators of both G_{ST} and G'_{ST} were calculated by simply substituting the H_S and H_T components of each statistic with their estimators calculated using equations 4 and 5 respectively. $D_{estChao}$ was calculated using the method described in (Chao *et al.*, 2008) (eqn 6 below). The formulae are as follows:

$\hat{H}_S$ (**Nei & Chesser, 1983**)

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

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

$\hat{H}_T$ (**Nei & Chesser, 1983**)

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

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

$D_{est(Chao)}$ (**Chao *et al.*, 2008; Jost, 2008**)

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

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

F_{ST} (**i.e. $\hat{\theta}$**) (**Weir & Cockerham, 1984**)

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

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

3.1.3 Bootstrapping

The variance of each statistic can be assessed using the bootstrapping method implemented in `diveRsity`. 95% confidence intervals are calculated by taking the upper and lower 2.5% sample quantiles form the bootstrapped parameter distribution.

For pairwise calculations carried out by the function, `fastDivPart`, a bias corrected 95% confidence interval is also calculated. This method accounts for the bias commonly associated with bootstrapping these types of statistics.

The method is basically a technique which re-centers the confidence interval around the initial parameter estimate. Say θ is our parameter of interest, $\hat{\theta}$ is out parameter estimate, and $\hat{\theta}^*$ is a vector of length, n, containing estimates of $\hat{\theta}$ for resampled data. The correction is carried out as follows:

$$\hat{\theta}^*_{[BC]} = \hat{\theta}^* - z0$$

Where $z0 = \frac{\sum \hat{\theta}^*}{n} - \hat{\theta}$.

The 2.5% and 97.5% quantile are then taken from the re-centered vector, $\hat{\theta}^*_{[BC]}$.

3.2   inCalc()

NOTE

This function was previously known as `in.calc`. This name has since been deprecated. Please use `inCalc` instead. `inCalc` allows the calculation of

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

$I_n(alleles)$ (**Rosenberg *et al.*, 2003**)

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

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

$I_n(locus)$ **(Rosenberg *et al.*, 2003)**

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

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

3.3   readGenepop()

NOTE

This function was previously known as `readGenepop.user`. This name has been deprecated as of version 1.4.4.

Although the `readGenepop` function is used extensively in both `divPart` and `inCalc`, its complexity is well hidden from general useRs. However, it has been included in `diveRsity` as a usable function for more experienced useRs, who may find it useful for data exploration and the development of analysis methods. As of version 1.3.0, this function is also implemented for use with the function `corPlot`. The function `readGenepop` returns up to 18 distinct `variables` (described in detail below), some of which have particularly

complex structures. Although this manual provides basic summaries of each returned `variable`, for the function to be useful, useRs are advised to explore the individual objects. This can be done using functions such as `str`, `names` and `typeof`.

3.4   corPlot()

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

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

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

The requirement to validate the use of certain marker types to infer demography is particularly important given that such information is often used to inform conservation and management strategies. It has been shown extensively in the literature that the relationship between F_{ST} and Nm in equation 10 breaks down if other evolutionary forces are strong (e.g. (Whitlock & McCauley, 1999)). For example if migration rate (m) is not $\gg$ than mutation

rate (μ), then $F_{ST} \neq \frac{1}{4Nm+1}$, and the quantity Nm cannot be accurately derived.

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

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

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

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

Thus, because of the negative dependence of F_{ST} on heterozygosity, and the positive dependence of heterozygosity on number of alleles, we can predict a

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

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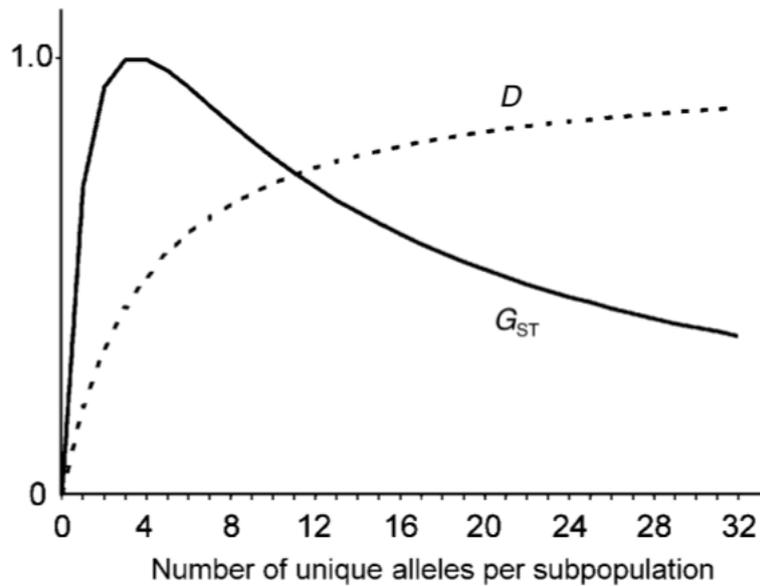
From this figure, it is clear that where the number of alleles at a locus is high (thus heterozygosity is high), G_{ST} is expected to be low. It is important to note that the negative relationship for G_{ST} and positive relationship for D_{Jost}, are complex, with multiple contributing factors. Thus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

3.5 difPlot()

New to v1.3.0
`difPlot` is yet another plotting function introduced to help useℝs easily visualise trends in their analysis results. Modern population genetic studies typically involve large numbers of population samples. It is often useful to know the pairwise relationships between each of these population samples. Due to the relationship between the number of sampled populations and the maximum number of possible pairwise comparisons, shown below, pin-pointing comparisons of interest can be very difficult.

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

$$N_{pw} = (1/2n)(n-1)$$
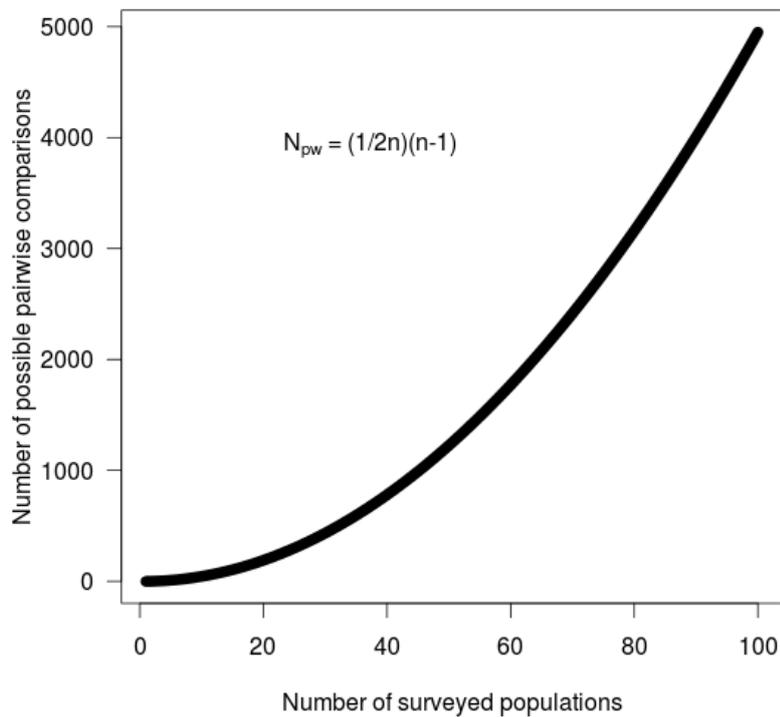
To overcome this problem, `difPlot` plots the pairwise values calculated by the function `divPart` using a diagonal matrix coupled with a colour gradient used to indicate the magnitude of a particular pairwise value. The function plots the estimated pairwise values for G_{ST}, θ, G'_{ST} and D_{Jost}.

3.6 chiCalc

New to v1.3.6
`chiCalc` allows the calculation of X^2 statistics for population genetic heterogeneity. The function contains a unique feature which allows users to exclude particular alleles if they are not observed frequently enough to be considered reliably. This feature allows a more conservative assessment of population genetic structure, but may results in a loss of power to detect actual differences.

3.7 divOnline

New to v1.3.6
`divOnline` is a simple function which allows users to launch a web app version of the `divPart` function. This function provides a less flexible but much more user friendly interface for the use of the `diveRsity` package. The web app was built using the `shiny` package from RStudio and Inc (RStudio & Inc., 2012).

3.8 fstOnly

New to v1.4.4

This function was written as a more RAM efficient way to calculate Weir & Cockerham's (1984) F_{ST} and F_{IT}, than `divPart`. The function should be particularly useful to users wishing to analyse large SNP data sets, where negative dependence on heterozygosity is not a concern.

3.9 divRatio

New to v1.4.4

This function introduces a new implementation of the method presented in Skrbinšek *et al.,* (2012), whose paper proposes the use of a well-characterised, 'yardstick' population sample to calculate a standardised diversity ratio for less well characterised population samples.

Interested users should see the original paper for a comprehensive description and justification of the method.

3.10 bigDivPart

New to v1.4.6

This function is the first in a series focused in the analysis of data sets containing large number of marker loci (e.g. RAD-seq derived SNPs). The function implements a more memory efficient programming technique (i.e.

extensive use of array data structures) to overcome some of the limitations associated with `divPart`. `bigDivPart` allows users to calculate all parameters calculated by `divPart` except via bootstrapping (e.g. locus and global G_{st}, G'_{st}, θ and D_{Jost} etc..).

3.11 microPlexer

New to v1.5.3
`microPlexer` is a simple function which allows users to launch a web application designed to aid in the development of microsatellite multiplex systems. The application allows users to flexibly organise microsatellite loci into PCR groups based on two main algorithms. The first, the *high-throughput* algorithm, aims to group as many loci into as few multiplex groups as possible, while ensuring user defined locus separation limits are observed. The second, the *balanced throughput* algorithm, organises loci into multiple multiplex groups with, roughly, the same number of loci. This algorithm is useful for minimising the possible risk of primer interactions during PCR. The web app was built using the `shiny` package from RStudio and Inc (RStudio & Inc., 2012).

3.12 arp2gen

New to v1.5.6 `arp2gen` is a small function allowing the conversion of *Arlequin* genotype files to *genepop* file format.

3.13 divMigrate

New to v1.5.7
`divMigrate` is an **experimental** function, allowing users to investigate and

plot migration pattern in microsatellite allele data. See `?divMigrate` for more details. The paper that the method is derived from is (Sundqvist *et al.*, 2013).

3.14   haploDiv

New to v1.6.1
`haploDiv` allows users to calculate Weir & Cockerham's (1984) θ from haploid genotype data. These data should be provided in the form of a two digit or three digit genepop file. The function essentially reads the haploid genotypes, *diploidizes* them (e.g. alleles coded as '01' will become '0101'), then the function `fastDivPart` is used to calculate F_{ST} estimates for various user defined levels. The function provides the facility to write a pairwise F_{ST} matrix to .txt file, as well as pairwise bootstrapped 95% confidence limits for each pairwise comparison.

4   Function Usage

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

4.1   divPart()

The general usage of this function is as follows:

```
> divPart(infile = NULL, outfile = NULL, gp = 3, pairwise = FALSE,
+         WC_Fst = FALSE, bs_locus = FALSE, bs_pairwise = FALSE,
+         bootstraps = 0, plot = FALSE, parallel = FALSE)
```

4.1.1   Arguments

|	
infile	Specifies the name of the *'genepop'* (Raymond & Rousset, 1995) file from which the statistics are to be calculated. This file can be in either the 3 digit of 2 digit format. The name must be a character string.
outfile	Allows useℝs to specify a prefix for an output folder. Name must a character string enclosed in either "" or ''.
gp	Specifies the digit format of the `infile`. Either 3 (default) or 2.
pairwise	A logical argument indicating whether pairwise matrices for each relevant statistic should be calculated. This feature can increase computation time for large number of population samples. Calculations will be made in parallel if the argument `parallel = TRUE`

`WC_Fst`	A logical indication as to whether Weir and Cockerham's, 1984 F-statistics should be calculated. This option will increase analysis time.
`bs_locus`	Gives useRs the option to bootstrap locus statistics. Results will be written to *.xlsx* workbook by default if the package `xlsx` is installed, and to a *.html* file if `plot = TRUE`. If `xlsx` is not installed, results will be written to *.txt* files.
`bs_pairwise`	Gives useRs the option to bootstrap statistics across all loci for each pairwise population comparison. Results will be written to a *.xlsx* file by default if the package `xlsx` is installed, and to a *.html* file if `plot = TRUE`. If `xlsx` is not installed, results will be written to *.txt* files.
`bootstraps`	Determines the number of bootstrap iterations to be carried out. The default value is `bootstraps = 0`, this is only valid when all bootstrap options are false. There is no upper limit on the number of bootstrap iterations, however very large numbers of bootstrap iterations for pairwise calculations ($>$ 1000) may take a long time to run for large data sets and may also lead to excessive RAM consumption. As an example, a test data set containing over 4000 individuals across 97 population samples typed for 15 microsatellite loci, took 1.5 days to complete on a Windows 7 ultimate 64bit machine with an Intel Core i5-2435M CPU @ 2.40GHz x 4.
`plot`	Optional interactive *.html* image files of the plotted bootstrap results for loci if `bs_locus = TRUE` and pairwise population comparisons if `bs_pairwise = TRUE` and the package `sendplot` is installed. The default option is `plot = FALSE`.
`parallel`	A logical argument specifying if computations should be run in parallel on all available CPU cores. If `parallel = TRUE`, batches of jobs will be distributed to all cores resulting in faster completion.

4.1.2 Returned values

Results returned by `divPart` vary depending on the argument options chosen. If the packages `xlsx` and `sendplot` are installed, results will be written to a single *.xlsx* workbook and *.png/.html* files providing `plot = TRUE`. Alternatively, if these packages are unavailable the plot option is no longer available. Results will be written to multiple *.txt* files, the number of which varies between three and five depending on the argument options chosen. An example screenshot of the *.xlsx* output file is shown below:
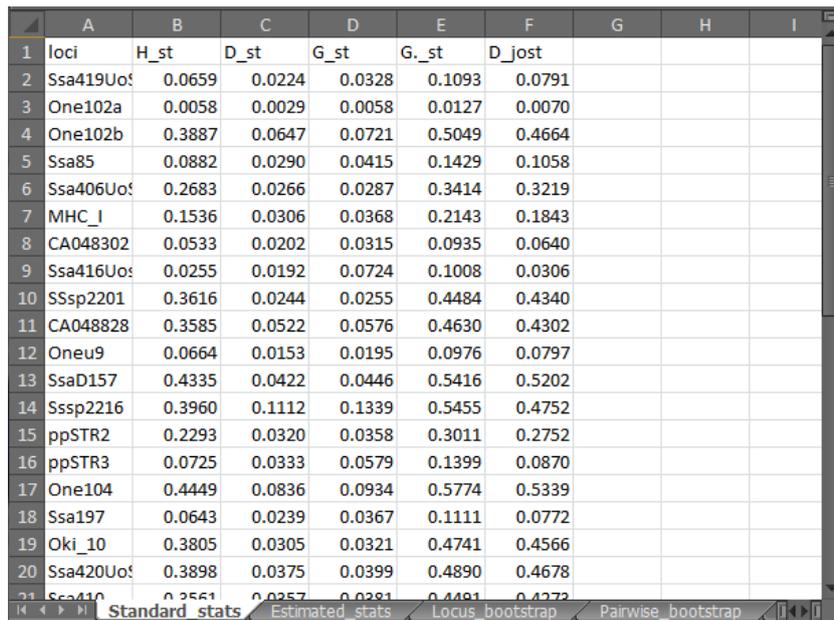
	A	B	C	D	E	F	G	H	I
1	loci	H_st	D_st	G_st	G._st	D_jost			
2	Ssa419Uo!	0.0659	0.0224	0.0328	0.1093	0.0791			
3	One102a	0.0058	0.0029	0.0058	0.0127	0.0070			
4	One102b	0.3887	0.0647	0.0721	0.5049	0.4664			
5	Ssa85	0.0882	0.0290	0.0415	0.1429	0.1058			
6	Ssa406Uo!	0.2683	0.0266	0.0287	0.3414	0.3219			
7	MHC_I	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	Oki_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	0.3561	0.0357	0.0381	0.4491	0.4273			

Standard_stats / Estimated_stats / Locus_bootstrap / Pairwise_bootstrap

Returned values cont.

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

Example of bootstrapped locus results plot
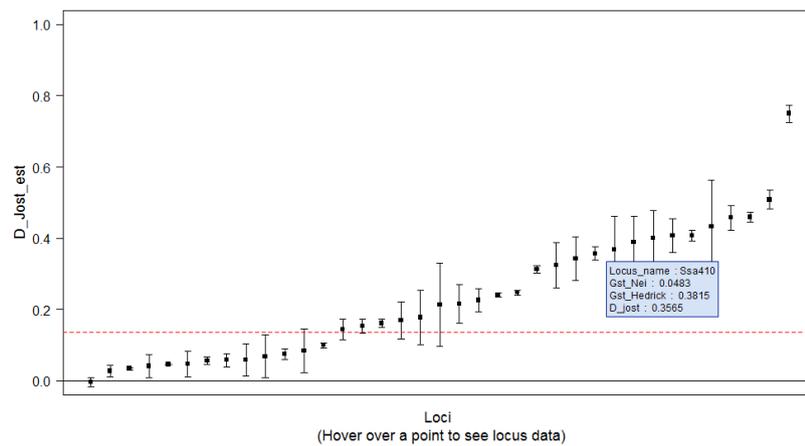
Example of bootstrapped

pairwise results plot
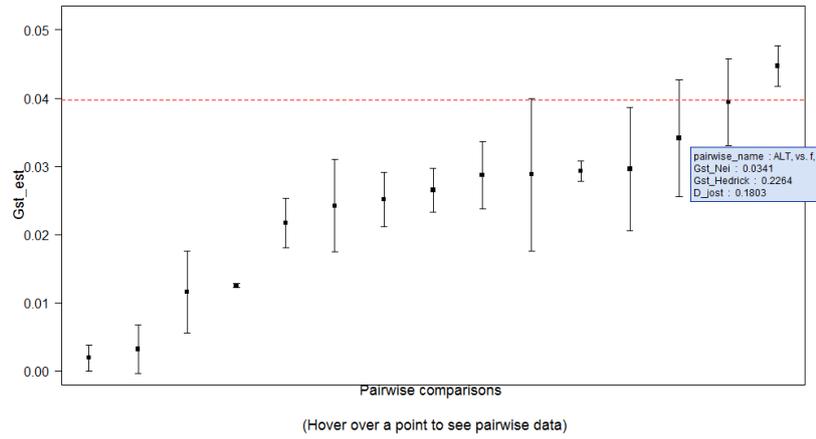
Returned values cont.

For useRs wishing to carry out post analysis manipulations, all results from `divPart` 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[divPart].txt* text file. The last row in this matrix represents statistics calculate across all population samples and loci.

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

loci

A list of locus names

H_st

Between subpopulation heterozygsity per locus

D_st

Absolute differentiation per locus (Nei, 1973)

G_st

F_st analogue for multiple alleles per locus (Nei, 1973)

G_hed_st

Hedrick's standardized "differentiation" per locus (Hedrick, 2005)

D_jost

Jost's true allelic differentiation per locus (Jost, 2008)

Returned values cont.

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

	Harmonic_N	H_st_est	D_st_est	G_st_est
Locus1	43.1218	0.0482	0.0160	0.0234
Locus2	43.5209	-0.0038	-0.0019	-0.0038
Locus3	43.6403	0.3610	0.0566	0.0629
Locus4	43.4476	0.0700	0.0225	0.0321
Locus5	42.7674	0.1998	0.0177	0.0191
Locus6	43.4476	0.1201	0.0228	0.0274
Locus7	43.4476	0.0382	0.0142	0.0221
Locus8	43.2566	0.0224	0.0168	0.0632
Locus9	43.0673	0.2703	0.0153	0.0160
Locus10	43.2469	0.3237	0.0439	0.0483
Global	NA	NA	NA	0.0397

	G_hed_st_est	D_Jost_est	Fst_WC	Fit_WC
Locus1	0.0799	0.0578	0.0257	0.0610
Locus2	-0.0084	-0.0046	-0.0042	-0.0518
Locus3	0.4688	0.4332	0.0745	0.0991
Locus4	0.1134	0.0840	0.0357	0.0555
Locus5	0.2542	0.2397	0.0222	0.0749
Locus6	0.1675	0.1441	0.0300	0.2250
Locus7	0.0670	0.0459	0.0258	0.0427
Locus8	0.0884	0.0268	0.0689	0.2529
Locus9	0.3352	0.3244	0.0189	0.0588
Locus10	0.4181	0.3885	0.0564	0.0986
Global	0.1806	0.1462	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 (Nei & Chesser, 1983)

D_st_est

Estimator of absolute differentiation (Nei & Chesser, 1983)

G_st_est

Nearly unbiased estimator of G_st (Nei & Chesser, 1983)

G_hed_st_est

Estimator of Hedrick's G'_st (Hedrick, 2005)

D_Jost_est

Estimator of Jost's D (Jost, 2008)

Fis_WC

Weir and Cockerham's inbreeding coefficient estimator (Weir & Cockerham, 1984)

Fst_WC

Weir and Cockerham's fixation index estimator (Weir & Cockerham, 1984)

Fit_WC

Weir and Cockerham's overall fixation index estimator (Weir & Cockerham, 1984)

Returned values cont.

$pairwise   A list of six (`WC_Fst = FALSE`) nine (`WC_Fst = TRUE`) ma-
                 trices containing pairwise diversity statistics without boot-
                 strapped confidence intervals.

```
[1] Gst


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



[1] G_hed_st


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



[1] D_Jost


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



[1] Gst_est
```

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

[1] G_hed_st_est

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

[1] D_Jost_est

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

[1] Fst_WC

```
          pop1,   pop2,   pop3, pop4,
pop1,       NA      NA      NA    NA
pop2, 0.0027      NA      NA    NA
pop3, 0.0647 0.0543      NA    NA
pop4, 0.0563 0.0478 0.0057    NA
```

[1] Fit_WC

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

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

Returned values cont.

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

[1] Gst

```
       Mean Lower_CI Upper_CI
Locus1 0.0342   0.0237   0.0464
Locus2 0.0153   0.0067   0.0282
Locus3 0.0824   0.0726   0.0925
global 0.0587   0.0579   0.0602
```

[1] G_hed_st

```
       Mean Lower_CI Upper_CI
Locus1 0.1156   0.0837   0.1559
Locus2 0.0330   0.0146   0.0601
Locus3 0.5472   0.5006   0.5756
global 0.2501   0.2468   0.2537
```

[1] D_Jost

```
       Mean Lower_CI Upper_CI
Locus1 0.0844   0.0614   0.1147
Locus2 0.0182   0.0080   0.0330
Locus3 0.5068   0.4615   0.5324
global 0.2033   0.2005   0.2059
```

[1] Gst_est

```
         Mean Lower_CI Upper_CI
Locus1 0.0248    0.0142   0.0371
Locus2 0.0057   -0.0030   0.0186
Locus3 0.0735    0.0636   0.0835
global 0.0492    0.0484   0.0507
```

[1] G_hed_st_est

```
         Mean Lower_CI Upper_CI
Locus1 0.0856    0.0513   0.1276
Locus2 0.0123   -0.0067   0.0402
Locus3 0.5163    0.4656   0.5486
global 0.2170    0.2139   0.2211
```

[1] D_Jost_est

```
         Mean Lower_CI Upper_CI
Locus1 0.0626    0.0377   0.0940
Locus2 0.0068   -0.0037   0.0221
Locus3 0.4783    0.4294   0.5074
global 0.1778    0.1746   0.1835
```

[1] Fst_WC

```
         Mean Lower_CI Upper_CI
Locus1 0.0277    0.0159   0.0418
Locus2 0.0076   -0.0023   0.0230
Locus3 0.0863    0.0754   0.0971
global 0.0569    0.0559   0.0587
```

[1] Fit_WC

```
          Mean Lower_CI Upper_CI
Locus1  0.0588    0.0395   0.0790
```

```
Locus2 -0.1175  -0.1617  -0.0771
Locus3  0.0990   0.0597   0.1610
global  0.1042   0.1020   0.1078
```

Returned values cont.

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

[1] Gst

```
                Mean Lower_CI Upper_CI
pop1, vs. pop2, 0.0138   0.0121   0.0163
pop1, vs. pop3, 0.0414   0.0388   0.0434
pop1, vs. pop4, 0.0407   0.0390   0.0421
pop5, vs. pop6, 0.0339   0.0325   0.0360
```

[1] G_hed_st

```
                Mean Lower_CI Upper_CI
pop1, vs. pop2, 0.0834   0.0733   0.0967
pop1, vs. pop3, 0.2570   0.2408   0.2722
pop1, vs. pop4, 0.2567   0.2500   0.2653
pop5, vs. pop6, 0.2257   0.2145   0.2391
```

[1] D_Jost

```
                Mean Lower_CI Upper_CI
pop1, vs. pop2, 0.0705   0.0620   0.0816
pop1, vs. pop3, 0.2250   0.2100   0.2394
pop1, vs. pop4, 0.2253   0.2198   0.2331
pop5, vs. pop6, 0.1984   0.1878   0.2105
```

[1] Gst_est

```
                 Mean Lower_CI Upper_CI
pop1, vs. pop2, 0.0079   0.0061   0.0104
pop1, vs. pop3, 0.0354   0.0327   0.0375
pop1, vs. pop4, 0.0354   0.0336   0.0369
pop5, vs. pop6, 0.0276   0.0264   0.0297


[1] G_hed_st_est


                 Mean Lower_CI Upper_CI
pop1, vs. pop2, 0.0495   0.0383   0.0639
pop1, vs. pop3, 0.2282   0.2107   0.2442
pop1, vs. pop4, 0.2308   0.2229   0.2404
pop5, vs. pop6, 0.1916   0.1812   0.2056


[1] D_Jost_est


                 Mean Lower_CI Upper_CI
pop1, vs. pop2, 0.0301   0.0186   0.0489
pop1, vs. pop3, 0.1902   0.1835   0.2004
pop1, vs. pop4, 0.1815   0.1800   0.1837
pop5, vs. pop6, 0.1517   0.1325   0.1695


[1] Fst_WC


                 Mean Lower_CI Upper_CI
pop1, vs. pop2, 0.0147   0.0111   0.0194
pop1, vs. pop3, 0.0673   0.0623   0.0712
pop1, vs. pop4, 0.0674   0.0643   0.0703
pop5, vs. pop6, 0.0529   0.0506   0.0571


[1] Fit_WC


                 Mean Lower_CI Upper_CI
pop1, vs. pop2, 0.1039   0.0919   0.1189
```

```
pop1, vs. pop3, 0.1205   0.1109   0.1258
pop1, vs. pop4, 0.1104   0.1050   0.1203
pop5, vs. pop6, 0.0751   0.0664   0.0863
```

4.2   inCalc()

The general usage of this function is as follows:

```
> inCalc(infile, outfile = NULL, gp = 3, bs_locus = FALSE,
+        bs_pairwise = FALSE, bootstraps = 0, plot = FALSE,
+        parallel = FALSE)
```

4.2.1   Arguments

infile
: Specifying the name of the *'genepop'* (Raymond & Rousset, 1995) file from which the statistics are to be calculated This file can be in either the 3 digit of 2 digit format. The name must be a character string.

outfile
: Allows useRs to specify a suffix for output folder and files. Name must a character string enclosed in either "" or ''.

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

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

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

Arguments cont.

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

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

parallel   A logical argument specifying if computations should be run in parallel on all available CPU cores. If `parallel = TRUE`, batches of jobs will be distributed to all cores resulting in faster completion.

4.2.2 Returned values

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

Example of bootstrapped locus I_n results

	A	B	C	D
1	Loci	Actual_In	Lower_95(	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	Oki_10	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	CA053293	0.022	0.0201	0.0239
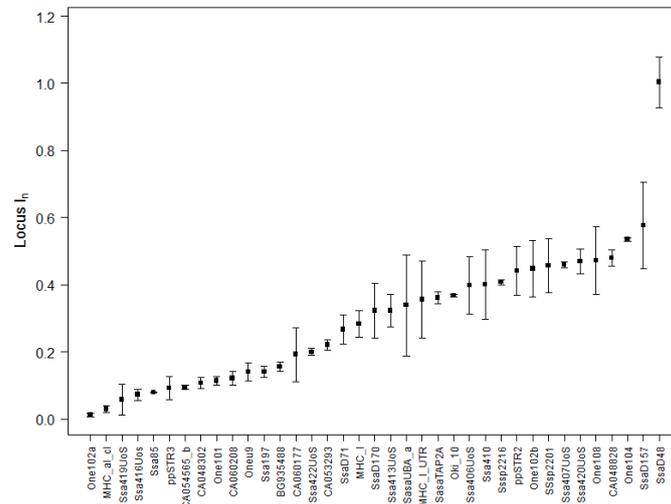
Overall_Bootstrap   Pairwise_bootstraps

Example of bootstrapped locus I_n results plot

Returned values cont.

For useRs wishing to carry out post analysis manipulations, all results from `inCalc` 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
Locus1	0.0036	0.0036	0.0144	0.004
Locus2	0.0095	0.0015	0.0013	
Locus3	0.0473	0.004	0.0098	0.0234
Locus4	0.0032	0.0029	0.0053	0.0135
Locus5	0.0111	0.0029	0.0042	0.0045
Locus6	0.0394	0.0379	0.0181	0.005
Locus7	0.0077	0.0131	0.0046	0.0087
Locus8	0.0157	0.0469	0.0054	0.0048
Locus9	0.0107	0.0075	0.0069	0.0054
Locus10	0.0038	0.0232	0.0091	0.0326

	Allele.5	Sum
Locus1	0.0178	0.0581
Locus2		0.0123
Locus3	0.027	0.4482
Locus4	0.0109	0.08
Locus5	0.0044	0.3983
Locus6	0.0352	0.2839
Locus7	0.0166	0.1068
Locus8		0.0728
Locus9	0.0081	0.4571
Locus10	0.0295	0.4799

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

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

```
           In Lower_95CI Upper_95CI
Locus1  0.0627     0.0434     0.0764
Locus2  0.0176     0.0092     0.0294
Locus3  0.4878     0.4797     0.4964
Locus4  0.0870     0.0675     0.1180
Locus5  0.4702     0.4545     0.4816
Locus6  0.3215     0.2931     0.3404
Locus7  0.1148     0.1033     0.1211
Locus8  0.0891     0.0706     0.1019
Locus9  0.5489     0.5307     0.5743
Locus10 0.5286     0.5136     0.5512
```

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	0.0337	0.0292	0.0381
Locus2	0.0210	0.0118	0.0314
Locus3	0.1081	0.1017	0.1131
Locus4	0.0600	0.0330	0.0832
Locus5	0.1298	0.1182	0.1449

[1] pop1, vs. pop3,

	In	Lower_95CI	Upper_95CI
Locus1	0.0340	0.0261	0.0408
Locus2	0.0156	0.0014	0.0330
Locus3	0.3344	0.3188	0.3437
Locus4	0.1161	0.0741	0.1617
Locus5	0.2973	0.2762	0.3292

[1] pop1, vs. pop4,

	In	Lower_95CI	Upper_95CI
Locus1	0.0277	0.0189	0.0380
Locus2	0.0084	0.0049	0.0122
Locus3	0.3277	0.3251	0.3316
Locus4	0.0500	0.0309	0.0816
Locus5	0.3122	0.2835	0.3289

[1] pop1, vs. pop5,

```
          In Lower_95CI Upper_95CI
Locus1 0.0697     0.0683     0.0715
Locus2 0.0117     0.0096     0.0133
Locus3 0.4010     0.3857     0.4228
Locus4 0.0731     0.0381     0.1109
Locus5 0.2869     0.2185     0.3562
```

[1] pop1, vs. pop6,

```
          In Lower_95CI Upper_95CI
Locus1 0.0286     0.0197     0.0422
Locus2 0.0163     0.0010     0.0280
Locus3 0.3524     0.3353     0.3747
Locus4 0.0380     0.0287     0.0446
Locus5 0.2543     0.2244     0.2764
```

4.3 readGenepop()

The general usage of `readGenepop` is:

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

4.3.1 Arguments

infile	Specifying the name of the *'genepop'* (Raymond & Rousset, 1995) file from which the statistics are to be calculated This file can be in either the 3 digit of 2 digit format. The name must be a character string.
gp	Specifies the digit format of the `infile`. Either 3 (default) or 2.
bootstrap	A logical argument indicating whether the `infile` should be sampled with replacement and all returned parameters calculated from this bootstrapped data.

4.3.2 Returned values

npops	The number of population samples in `infile`.
nloci	The number of loci in `infile`.
pop_alleles	A list of two matrices per population. Each matrix per population contains haploid allele designations.
pop_list	A list of matrices (n = `npops`) containing the diploid genotypes of individuals per locus.
loci_names	A character vector containing the names of loci from `infile`.

pop_pos	A numeric vector or the row index locations of the first individual per population in `infile`.
pop_sizes	A numeric vector of length `npops` containing the number of individuals per population sample in `infile`.
allele_names	A list of `npops` lists containing `nloci` character vectors of alleles names per locus. Useful for identifying unique alleles.
all_alleles	A list of `nloci` character vectors of all alleles observed across all population samples in `infile`.
allele_freq	A list containing `nloci` matrices containing allele frequencies per alleles per population sample.
raw_data	An unaltered data frame of `infile`.
loci_harm_N	A numeric vector of length `nloci`, containing the harmonic mean number of individuals genotyped per locus.
n_harmonic	A numeric value representing the harmonic mean of `npops`.
pop_names	A character vector containing a six character population sample name for each population in `infile` (the first six characters of the first individual).
indtyp	A list of length `nloci` containing character vectors of length `npops`, indicating the number of individuals per population sample typed at each locus.
nalleles	A vector of the total number of alleles observed at each locus.
bs_file	A dataframe/genpop object of bootstrapped data. Returned if `bootstrap = TRUE`.
obs_all...	A list of matrices of the observed number of allele occurrences per population.

4.4   corPlot()

The general usage of `corPlot` is:

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

4.4.1   Arguments

x   The object returned by the function `readGenepop`.

y   The object returned by the function `divPart`.

4.4.2   Returned values

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

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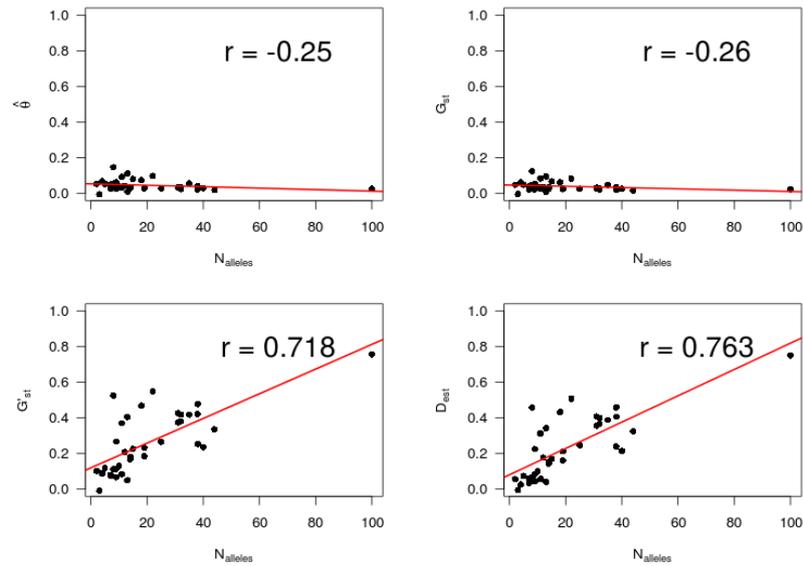
The plot depicts the relationship between the estimated statistics calculated by `divPart` and the number of alleles per locus. Lines represents the line of best fit. Pearson's product-moment correlation coefficient is also provided.

4.5 difPlot()

The general usage of `difPlot` is:

```
> difPlot(x, outfile = NULL, interactive = FALSE)
```

4.5.1 Arguments

x   The object returned by the function `divPart`.

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

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

4.5.2 Returned values

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

Returned plot from the function difPlot when interactive = FALSE
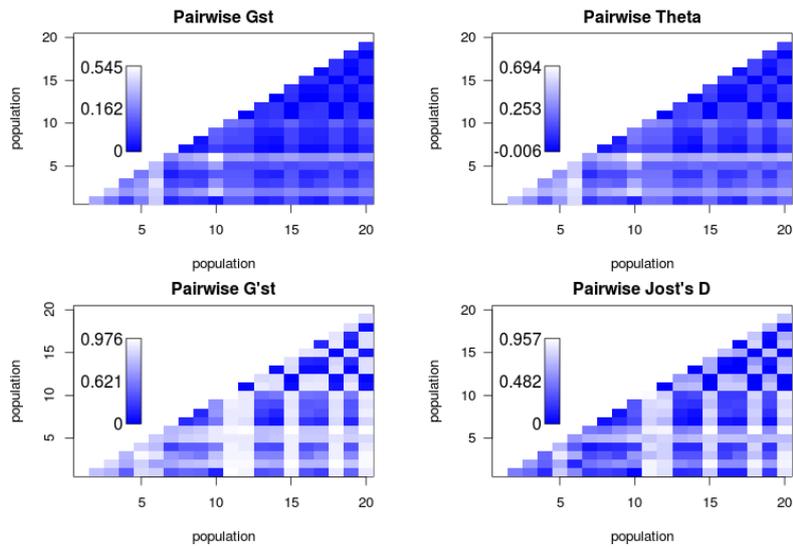
One of the returned plots from the function difPlot when interactive=TRUE

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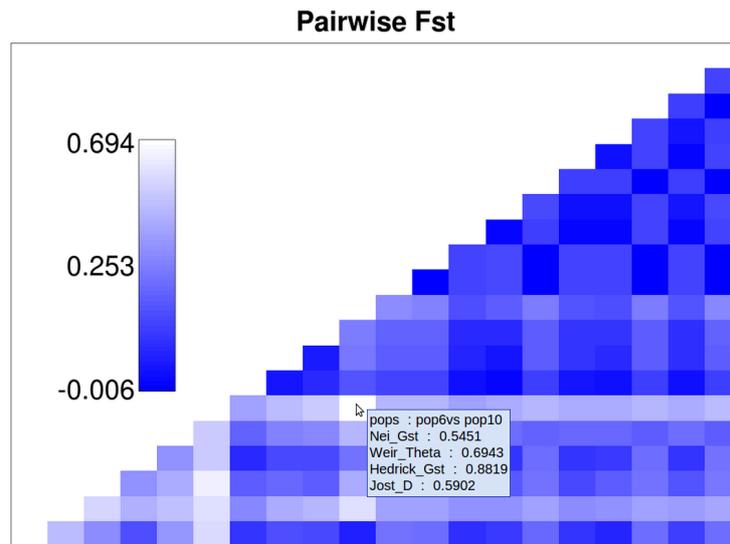
As can be seen, the plots produced when `interactive = TRUE` are much more useful than when `interactive = FALSE`, due to useRs ability to identify population comparisons of interest. These plots contain tool-tip information, courtesy of the `sendplot` package.

4.6 chiCalc

The general usage of `chiCalc` is:

```
> chiCalc(infile = NULL, outfile = NULL, gp = 3, minFreq = NULL)
```

4.6.1 Arguments

infile
: Specifying the name of the *'genepop'* (Raymond & Rousset, 1995) file from which the statistics are to be calculated This file can be in either the 3 digit of 2 digit format. The name must be a character string.

outfile
: A character string specifying the name given to an output file, containing analysis results. If this argument is passed as `NULL`, no file will be written.

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

minFreq
: A threshold minimum value or vector of values, below which alleles are not included in the analysis.

4.6.2 Returned values

chi table
: A character matrix containing locus chi-square values, degrees of freedom, p.values and significance indicators, as well as overall values.

4.7 divOnline

The general usage of `divOnline` is:

```
> divOnline()
```

By executing the above command, a web browser (system default) will open with the divOnline application running. Users can read file from their system into the app and choose many of the analysis options. Most analysis results can be downloaded to *.txt* files.

4.8 fstOnly

The general usage of `fstOnly` is:

```
> fstOnly(infile = NULL, outfile = NULL, gp = 3, bs_locus = FALSE,
+         bs_pairwise = FALSE, bootstraps = 0, parallel = FALSE)
```

4.8.1 Arguments

infile
:   Specifying the name of the *'genepop'* (Raymond & Rousset, 1995) file from which the statistics are to be calculated This file can be in either the 3 digit of 2 digit format. The name must be a character string.

outfile
:   Allows useRs to specify a suffix for output folder and files. Name must a character string enclosed in either "" or ''.

gp	Specifies the digit format of the `infile`. Either 3 (default) or 2.
bs_locus	Gives useRs the option to bootstrap locus statistics. Results will be written to *.xlsx* file by default if the package `xlsx` is installed, and to a *.png* file if `plot=TRUE`. If `xlsx` is not installed, results will be written to *.txt* files.
bs_pairwise	Gives useRs the option to bootstrap statistics across all loci for each pairwise population comparison. Results will be written to a *.xlsx* file by default if the package `xlsx` is installed. If `xlsx` is not installed, results will be written to *.txt* files.
bootstraps	Determines the number of bootstrap iterations to be carried out. The default value is `bootstraps = 0`, this is only valid when all bootstrap options are false. There is no upper limit on the number of bootstrap iterations, however very large numbers of bootstrap iterations for pairwise calculations ($>$ 1000) may take a long time to run for large data sets.
parallel	A logical argument specifying if computations should be run in parallel on all available CPU cores. If `parallel = TRUE`, batches of jobs will be distributed to all cores resulting in faster completion.

4.8.2 Returned values

locus	A list contain two matrices, F_{ST} and F_{IT}. Each matrix contains the actual calculated statistic along with their respective 95% confidence intervals per locus, as well as a global estimate across all population samples and loci. This result is only returned if `bs_locus` = TRUE.
pairwise	A list contain two matrices, F_{ST} and F_{IT}. Each matrix contain the actual and respective 95% confidence intervals across loci for each pairwise population comparison. This result is only returned when `bs_pairwise` = TRUE.

64

4.9 divRatio

The general usage of `divRatio` is:

```
> divRatio(infile = NULL, outfile = NULL, gp = 3, pop_stats = NULL,
+          refPos = NULL, bootstraps = 1000, parallel = FALSE)
```

4.9.1 Arguments

infile
: A character string argument specifying the name of either a 3 digit or 2 digit genepop file containing the raw genotypes of at least the reference population sample.

outfile
: A character string specifying a prefix name for an automatically generated results folder, to which results file will be written.

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

pop_stats
: A character string indicating the name of the population statistics data frame file. This argument is required if only raw data for the reference population are give in `infile`. The data frame should be structured in a specific way. An example can be seen by typing `data(pop_stats)` into the console. The validloci column is only required if mean allelic richness and expected heterozygosity for populations of interest have been calculated from loci for which data is not present in the reference population. This column should contain a single character string of common loci between each population sample and the reference population sample.

refPos
: A numeric argument specifying the position of the reference population in `infile`. The argument is only valid when raw genotype data has been provided for the reference population sample and all other populations of interest and `pop_stats` is NULL.

bootstraps   Specifies the number of times the reference population should
                  be resampled when calculating the sample size standardised
                  allelic richness and expected heterozygosity for calculating
                  the diversity ratios. The larger the number of bootstraps
                  the longer the analysis will take to run. As an indication of
                  runtime, running `divRatio` on the `Big_data` data set (type
                  `?Big_data` for details), takes 10min 42s on a Toshiba Satel-
                  lite R830 with 6GB RAM, and an Intel Core i5 - 2435M
                  CPU running Linux.

parallel   A logical argument indicating whether the analysis should
                  make use of all available cores on the users system.

4.9.2   Returned values

All results will be written to a user defined folder, providing an argument is
passed for 'outfile'. Results will be written to `.xlsx` files if the package `xlsx`
and its dependencies are installed, or a `.txt` file otherwise.

A data frame containing the following variables is also returned to the `R`
console:

pop   The names of each population of interest, including the ref-
                  erence population.

n   The sample size of each population

alr   Mean allelic richness across loci

alrSE   The standard error of the allelic richness across loci

He   Mean expected heterozygosity across loci

HeSE   Standard error of expected heterozygosity across loci

alrRatio   The ratio of the allelic richness of the subject population
                  sample and the sample size standardised reference popula-
                  tion allelic richness

`alrSERation`	The standard error of divisions for the allelic richness ratio
`heRatio`	The ratio of expected heterozygosity between the standardised reference population sample and subject population samples
`heSEratio`	The standard error of divisions for the expected heterozygosity ratio

4.10   bigDivPart()

The general usage of this function is as follows:

```
> bigDivPart(infile = NULL, outfile = NULL, WC_Fst = FALSE,
+            format = NULL)
```

4.10.1   Arguments

`infile`	Specifies the name of the *'genepop'* (Raymond & Rousset, 1995) file from which the statistics are to be calculated. This file can be in either the 3 digit of 2 digit format. The name must be a character string.
`outfile`	Allows useRs to specify a prefix for an output folder. Name must a character string enclosed in either "" or ''.
`WC_Fst`	A logical indication as to whether Weir and Cockerham's, 1984 F-statistics should be calculated. This option will increase analysis time.
`format`	A character string specifying the preferred output format for calculated results. The arguments `txt` or `xlsx` are valid when `outfile` is not NULL.

4.10.2 Returned values

standard   See `divPart` description for details.

estimates   See `divPart` description for details.

4.11 microPlexer

The general usage of this function is as follows:

```
> microPlexer()
```

By typing the above command into the `R` console, a web application will be launched in the system's default internet browser. At initiation, the application appears as the figure below:

After uploading input data and making the desired parameter selections, multiplex group plots will be displayed as below. These plots can be downloaded to a single .PDF workbook for further inspection.
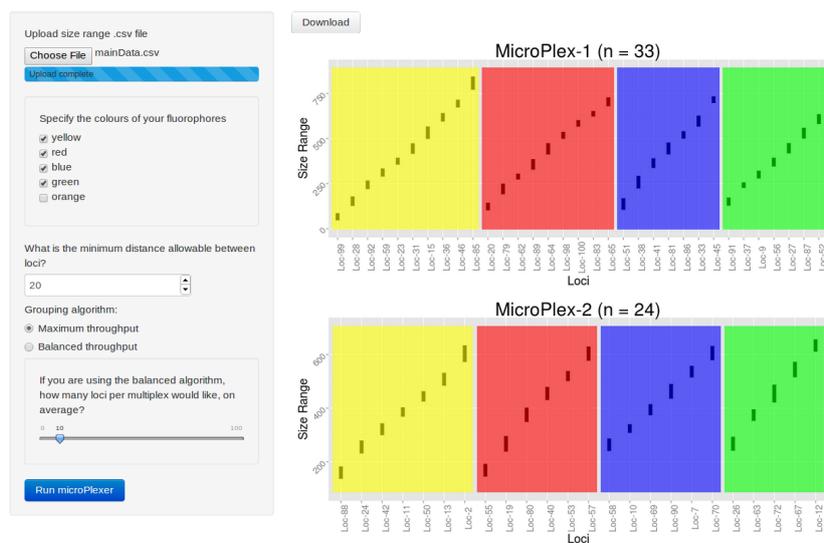
4.12 arp2gen

The general usage of this function is as follows:

```
> arp2gen(infile)
```

Where `infile` is a character string pointing to an *Arlequin* genotype file. The infile argument can simply be the file name if the file is located in the current working directory, otherwise an absolute/relative path must be given.

4.13 divMigrate

The general usage of this function is as follows:

```
> divMigrate(infile = NULL, stat = c("gst", "d_jost"))
```

Where `infile` is either an file name or a path to a file. The file should be in either the Arlequin genotype file format (.arp) or the genepop file format used in other `diveRsity` functions. The argument `stat` determines which statistic (G_{ST} or D_{Jost}) should be used to calculate the relative migration matrix.

4.14 haploDiv

The general usage of this function is as follows:

```
> haploDiv(infile = NULL, outfile = NULL, pairwise = FALSE,
+         bootstraps = 0)
```

Where `infile` is the file name or directory location of a haploid genotype genepop file, `outfile` is the prefix name to be used for output .txt files, `pairwise` indicates whether a pairwise matrix of Weir & Cockerham's θ should be calculated, and `bootstraps` indicates whether bootstrapped 95% confidence limits should be estimated for each pairwise comparison.

Details

If the `outfile` argument is `NULL`, no results will be written to file. All results are returned to the `R` workspace. To calculate 95% confidence limits, `bootstraps` must be > 0, and `pairwise` must be `TRUE`.

5 Examples

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

5.1 divPart

This example is specific to the function `divPart`. 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 `divPart`

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

```
> div_results <- divPart(infile = Test_data, outfile = "Test",
+                        gp = 3, pairwise = TRUE,
+                        WC_Fst = TRUE, 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 '*[divPart].xlsx*' (if `xlsx` is installed), or four *.txt* files named, '*Standard-stats[divPart].txt*', '*Estimated-stats[divPart].txt*', '*Locus-bootstrap[divPart].txt*' and '*Pairwise-bootstrap[divPart].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 `divPart`, 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"
[4] "meanPairwise" "bs_locus"    "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 `divPart`, 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] "Fst_WC"     "Fit_WC"
```

Each of the named objects within `test_results$bs_locus` are known to be
matrices from above. This means that we can use matrix indexing to access
any of the information within any of the matrices. In R, to access a specific
value within a matrix, we only need to know the row and column that the
value is in. If we wanted to access a value that lies in the 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_results$bs_locus$Gst`, we would use the following code.

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

```
         Mean Lower_CI Upper_CI
Locus1  0.0342  0.0237   0.0464
Locus2  0.0153  0.0067   0.0282
Locus3  0.0824  0.0726   0.0925
Locus4  0.0479  0.0438   0.0506
Locus5  0.0408  0.0368   0.0458
Locus6  0.0475  0.0365   0.0537
Locus7  0.0408  0.0304   0.0501
Locus8  0.0941  0.0502   0.1249
Locus9  0.0321  0.0302   0.0353
Locus10 0.0705  0.0667   0.0738
```

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$bs_locus$Gst`, we would use:

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

The other values returned by `divPart` 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   inCalc

This example is specific to the function `inCalc`. 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 `inCalc`

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

```
> in_results <- inCalc (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[inCalc].txt*', '*Overall-bootstrap[inCalc].txt*' and '*Pairwise-bootstrap[inCalc].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 `inCalc`, 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 `inCalc`, it is known that `PW_bootstrap`
is indeed a list of matrices of bootstrapped locus results for each pairwise
comparison. To find the names of the matrices within `PW_bootstraps`, use:

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

```
 [1] "pop1, vs. pop2," "pop1, vs. pop3,"
 [3] "pop1, vs. pop4," "pop1, vs. pop5,"
 [5] "pop1, vs. pop6," "pop2, vs. pop3,"
 [7] "pop2, vs. pop4," "pop2, vs. pop5,"
 [9] "pop2, vs. pop6," "pop3, vs. pop4,"
[11] "pop3, vs. pop5," "pop3, vs. pop6,"
[13] "pop4, vs. pop5," "pop4, vs. pop6,"
[15] "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 0.0337    0.0292     0.0381
Locus2 0.0210    0.0118     0.0314
Locus3 0.1081    0.1017     0.1131
```

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 `inCalc` 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

This example is specific to the function `readGenepop`. 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`

To run `readGenepop` without producing a bootstrap file, use:

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

5.3.4   Accessing your results within the R session

The `readGenepop` 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"
 [3] "pop_alleles"    "pop_list"
 [5] "loci_names"     "pop_pos"
 [7] "pop_sizes"      "allele_names"
 [9] "all_alleles"    "allele_freq"
[11] "raw_data"       "loci_harm_N"
[13] "n_harmonic"     "pop_names"
[15] "indtyp"         "nalleles"
[17] "obs_allele_num"
```

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

5.3.5 Applications for `readGenepop`

`readGenepop` 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` 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` returns an object `pop_alleles` which contains npops x 2 matrices. Each matrix contains a haploid genotype per individual per locus, and every two matrices correspond to a single population sample. For example matrices 1 and 2 correspond to population sample 1, matrices 3 and 4 correspond to population sample 2 and so on. Using this object, it is possible to plot the allele size distribution to assess it allele fragments fit say the single step mutation model (SSM).

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

From this figure we could conclude that locus 18 in population 1 is likely to

follow SSM given that allele size increases in a generally regular fashion. Any gaps are also a multiple of the repeat motif length.

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

5.3.6 A hypothetical example

This example is for illustrative purposes.

Say for some reason, we were interested in assessing the sampling properties of the number of alleles at a particular locus, `readGenepop` 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 we can fill the matrix
> bs<-10
> for(i in 1:bs){
+     # first produce a bootstrap file
+
+     x <- readGenepop(infile = Test_data, gp = 3,
+                   bootstrap = TRUE)
+
+     # Now record the number of alleles at each locus
+
```

```
+     num_all[i, ] <- x$nalleles
+ }
> # Now we can use this data to calculate the mean
> # number of alleles per locus as well at their
> # 95% confidence intervals
>
> mean_num <- colMeans(num_all)
> lower<-vector()
> upper<-vector()
> for(i in 1:ncol(num_all)){
+     lower[i] <- mean_num[i] - (1.96 * sd(num_all[,i]))
+     upper[i] <- mean_num[i] + (1.96 * sd(num_all[,i]))
+ }
> # Now we can create a data frame of these results
>
> bs_res <- data.frame(mean_num, lower, upper)
> bs_res[1:10,]
```

	mean_num	lower	upper
1	5.8	4.560387	7.039613
2	3.0	3.000000	3.000000
3	17.7	16.753230	18.646770
4	7.3	5.027376	9.572624
5	35.2	32.307570	38.092430
6	14.0	14.000000	14.000000
7	8.3	6.977101	9.622899
8	4.0	4.000000	4.000000
9	41.4	39.506460	43.293540
10	32.3	30.223674	34.376326

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

5.4 Running divPart in batch (using parallel)

Application of batch analyses

Often, for a number of reasons, it may be necessary to analyse many separate genepop file. Simulation studies and Approximate Bayesian Computation (ABC) are two examples of where this is commonly done. Below is a hypothetical example, demonstrating how this could be done using `diveRsity`

The hypothetical experiment

Imagine that we are interested in measuring some group of parameters for a group of populations simulated under various evolutionary models. In this example we would like to estimate global G_{st}, θ, G'_{st} and D_{Jost} for a group of population samples evolved under 10 distinct models. However, because we are interested in comparing the distributions of each of these parameters under each of the 10 models, we have replicated each simulation 1000 times (giving a total of 10,000 genepop files to be analysed). Helpfully, our 1000 genepop files per evolutionary model have been organised into 10 separate folders (1 per model). The directory tree might look something like this:

```
~/simulations
├── sim1
│       rep1.gen
│       rep2.gen
│       ...
│       ...
│       rep1000.gen
├── sim2
│       rep1.gen
│       rep2.gen
│       ...
│       ...
│       rep1000.gen
├── sim3
│       rep1.gen
│       rep2.gen
│       ...
│       ...
│       rep1000.gen
```

The workflow to analyse these files in parallel (say on 10 CPUs) is as follows:

- Determine the names and locations of all files to be analysed
- Set up the CPU cluster for all R processes (using the `doParallel` & `parallel` packages).
- Pipe these file names to the `divPart` function and return the parameters required (i.e. global G_{st}, θ, G'_{st} & D_{Jost}).
- Return the results in a convenient format to allow for down stream investigations.

The annotated code is below:

```
>  # load the diveRsity package
> library("diveRsity")
> # We can specify the names of our simulation folders in two ways
>
> # manually
> fold_names <- paste("sim", 1:10, sep = "")
> or
> # automatically (when there is only a single level below the
> # top directory)
> fold_names <- list.dirs(full.names = TRUE, recursive = FALSE)
> # Now we can determine the names of all genepop files in each folder
>
> file_names <- lapply(fold_names, function(x){
+   files <- dir(path = paste(x, "/", sep = ""), pattern = "*.gen",
+                full.names = TRUE)
+   return(files) })
> # file_names will be a list of length 10. Each element will contain
> # the names of all .gen files within the respective simulation folder
>
> # Before we are ready to run the main analyses, we should set up
> # the parallel environment
>
> # load the doParallel package
> library("doParallel")
> # set up a cluster of 10 CPUs (one for each batch of files)
> cl <- makeCluster(10)
> # Export the 'divPart' function to the cluster cores
> clusterExport(cl, "divPart", envir = environment())
> # Now we can run the main analyses
>
> results <- parLapply(cl, file_names, function(x){
+   sim_res <- sapply(x, function(y){
+     out <- divPart(infile = y, gp = 3, WC_Fst = TRUE)
+     return(out$estimate[nrow(out$estimate), 4:7])
+   })
+   return(t(sim_res))  # transpose sim_res
+ })
>
> # This will generate a list (of length 10), with each element
> # containing a matrix of 1000 rows (1 per file) and 4 columns
> # (1 for each diversity statistic)
```

```
>
> # Example of output for simulation 1
> # G_st_est         G_hed_st_est    D_Jost_est      Fst_WC
> # 0.3905           0.8938          0.8256          0.4010
> # 0.5519           0.8719          0.6986          0.6031
> # 0.5924           0.8880          0.7092          0.6096
> # ...              ...             ...             ...
> # ...              ...             ...             ...
>
> # these results could then be piped to further analyses or
> # visualisation tools
```

References

Chao, A., Jost, L., Chiang, S., Jiang, Y.H. & Chazdon, R.L. (2008) A two-stage probabilistic approach to multiple-community similarity indices. *Biometrics*, **64**, 1178–1186.

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

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

Hedrick, P. (2005) A standardized genetic differentiation measure. *Evolution*, **59**, 1633–1638.

Jost, L. (2008) G_{ST} and its relatives do not measure differentiation. *Molecular Ecology*, **17**, 4015–4026.

Keenan, K., McGinnity, P., Cross, T.F., Crozier, W.W. & Prodĭhl, P.A. (2013) diveRsity: An R package for the estimation and exploration of population genetics parameters and their associated errors. *Methods in Ecology and Evolution*, pp. n/a–n/a.

Lemon, J. (2006) Plotrix: a package in the red light district of r. *R News*, **6**, 8–12.

Nei, M. (1973) Analysis of gene diversity in subdivided populations. *Proceedings of the National Academy of Sciences*, **70**, 3321–3323.

Nei, M. & Chesser, R. (1983) Estimation of fixation indices and gene diversities. *Annals of Human Genetics*, **47**, 253–259.

O'Reilly, P.T., Canino, M.F., Bailey, K.M. & Bentzen, P. (2004) Inverse relationship between F and microsatellite polymorphism in the marine fish, walleye pollock (Theragra chalcogramma): implications for resolving weak population structure. *Molecular ecology*, **13**, 1799–814.

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

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

Raymond, M. & Rousset, F. (1995) GENEPOP (version 1.2): population genetics software for exact tests and ecumenicism. *Journal of Heredity*, **86**, 248.

Revolution Analytics (2012a) *doParallel: Foreach parallel adaptor for the parallel package.* R package version 1.0.1.

Revolution Analytics (2012b) *foreach: Foreach looping construct for R.* R package version 1.4.0.

Revolution Analytics (2012c) *iterators: Iterator construct for R.* R package version 1.0.6.

Rosenberg, N.A., Li, L.M., Ward, R. & Pritchard, J.K. (2003) Informativeness of genetic markers for inference of ancestry. *The American Journal of Human Genetics*, **73**, 1402–1422.

RStudio & Inc. (2012) *shiny: Web Application Framework for R.* R package version 0.2.3.

Skrbinšek, T., Jelenčič, M., Waits, L., Potočnik, H., Kos, I. & Trontelj, P. (2012) Using a reference population yardstick to calibrate and compare genetic diversity reported in different studies: an example from the brown bear. *Heredity*, **109**, 299–305.

Sundqvist, L., Zackrisson, M. & Kleinhans, D. (2013) Directional genetic differentiation and asymmetric migration.

Weir, B. & Cockerham, C. (1984) Estimating F-statistics for the analysis of population structure. *Evolution*, **38**, 1358–1370.

Whitlock, M. & McCauley, D. (1999) Indirect measures of gene flow and migration: $F_{ST} \neq 1/(4Nm + 1)$. *Heredity*, **82**, 117–25.

6 Reproducibility

```
R Under development (unstable) (2016-12-06 r71735)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Debian GNU/Linux stretch/sid

attached base packages:
[1] stats      graphics  grDevices utils
[5] datasets   methods   base

other attached packages:
[1] diveRsity_1.9.89

loaded via a namespace (and not attached):
 [1] splines_3.4.0        ellipse_0.3-8
 [3] network_1.13.0       gtools_3.5.0
 [5] Formula_1.2-1        shiny_0.14.2
 [7] assertthat_0.1       stats4_3.4.0
 [9] latticeExtra_0.6-28  d3Network_0.5.2.1
[11] pbivnorm_0.6.0       lattice_0.20-34
[13] quadprog_1.5-5       digest_0.6.10
[15] RColorBrewer_1.1-2   ggm_2.3
[17] minqa_1.2.4          colorspace_1.3-1
[19] httpuv_1.3.3         htmltools_0.3.5
[21] Matrix_1.2-7.1       plyr_1.8.4
[23] psych_1.6.9          xtable_1.8-2
[25] corpcor_1.6.8        scales_0.4.1
[27] glasso_1.8           sna_2.4
[29] whisker_0.3-2        jpeg_0.1-8
[31] fdrtool_1.2.15       lme4_1.1-12
[33] huge_1.2.7           arm_1.9-3
[35] htmlTable_1.7        tibble_1.2
[37] ggplot2_2.2.0        nnet_7.3-12
[39] lazyeval_0.2.0       mnormt_1.5-5
[41] mime_0.5             survival_2.40-1
[43] magrittr_1.5         statnet.common_3.3.0
[45] nlme_3.1-128         MASS_7.3-45
[47] foreign_0.8-67       tools_3.4.0
[49] data.table_1.10.0    stringr_1.1.0
```

```
[51] munsell_0.4.3          cluster_2.0.5
[53] compiler_3.4.0         sem_3.1-8
[55] grid_3.4.0             nloptr_1.0.4
[57] rjson_0.2.15           igraph_1.0.1
[59] lavaan_0.5-22          boot_1.3-18
[61] mi_1.0                 gtable_0.2.0
[63] abind_1.4-5            reshape2_1.4.2
[65] qgraph_1.4.1           R6_2.2.0
[67] gridExtra_2.2.1        knitr_1.15.1
[69] Hmisc_4.0-0            stringi_1.1.2
[71] matrixcalc_1.0-3       parallel_3.4.0
[73] Rcpp_0.12.8            rpart_4.1-10
[75] acepack_1.4.1          png_0.1-7
[77] coda_0.18-1
```

7   Acknowledgements