KGD: Software for GBS-based relationship calculations v1.4.0

Document: KGDManual.pdf
Author: Ken Dodds
Address: Invermay Agricultural Centre, Puddle Alley, Private Bag 50034, Mosgiel 9053, New Zealand
Email: ken.dodds at agresearch.co.nz
Date: 28 October 2025

Contents

Background

R code is available for the analysis of genotyping-by-sequencing (GBS) data, primarily to construct a genomic relationship matrix (‘G matrix’) for the genotyped individuals, with a focus on the KGD (Kinship using GBS with Depth adjustment) method of Dodds *et al*. (2015). Collectively this software is referred to as the KGD software. The code can be used on its own or incorporated into other R programs. There are QC tools (primarily graphical output), relationship estimation tools, pedigree verification tools and pedigree ‘mix and match’ tools. The latter two operations require additional input information about the samples genotyped. There are also tools implementing some methods for population genetics that are currently under development.

In this document, 'Individual' or 'sample' generally refers to the genotyping unit (possibly combined, if the same individual or sample is genotyped multiple times). Familial relationships are given the labels 'Father', 'Mother' and 'Offspring' (as appropriate).

Many of the methods used are as described in Dodds *et al*. (2015), Bilton *et al*. (2019) and Dodds *et al*. (2019). Unless specified otherwise, relatedness estimates in this documentation refer to those using the 'G5' method of Dodds *et al*. (2015).

Program structure

There are separate analysis program files, the first (GBS-Chip-Gmatrix.R) is for genotype QC and relationship matrix construction while there are also programs for pedigree verification and/or assignment, using the relatedness estimates (GBSPedAssign.R) and for population genetics analyses (GBS-PopGen.R). These programs can be invoked from another program file (using the *source* command), or users can insert all or parts of these programs into their own code. For the purposes of this documentation, it is assumed the first method is used, with calling program named GBSRun.R. The software was initially written to calculate a genomic relationship matrix, and this achieved with the function *calcG*. If any of these programs (or relevant functions) are being called multiple times in an R session, care should be taken to make sure any default values still apply for the subsequent call(s), otherwise they should be removed (rm) or set specifically.

Version 1 major changes.

Use of depth.orig

Version 1 no longer creates or used the matrix *depth.orig*. The matrix *depth* is no longer modified by the main program. If users have referred to *depth.orig* with version 0, they should replace that with *depth* when using version 1.

GBSPedAssign.R

This no longer needs to be sourced each time an analysis is required, but can be invoked with *GBSPed*(). If *functions.only* is FALSE when sourcing GBSPedAssign.R, then the code is run and results returned to the list *PedResults*.

The main results objects that were previously (in version 0) available in the global environment after the pedigree analysis (i.e. one or more of *pedinfo*, *FatherMatches*, *MotherMatches* and *BothMatches*) are now returned (with the same names) in the list returned from *GBSPed*(). Any code that referred to these objects when using version 0 will now need to obtained them from the *GBSPed*() output object.

Calling program (GBSRun.R)

Example code for using KGD is giving in a calling program, GBSRun.R. The R *source* command is used to invoke the GBS-Chip-Gmatrix.R code. This code will (optionally, see *functions.only*) read data, run QC procedures and define functions, e.g. to read data (*readGBS*), perform QC (*GBSsummary*) or calculate the genomic relationship matrix (*calcG*). Variables that can be set for using in the main program (i.e., QC and relatedness estimation) are shown in Table 1. See other sections (pedigree analysis, population genetics) for details on variables and commands relevant to those analysis.

Table 1 Variables that can be set in the calling program to control analyses.

Variable	Description

genofile	Name (including path) of the genotype file. Default value is "HapMap.hmc.txt".
gform	Type of genotype file. Default is "uneak"; other options are "Tassel", "TagDigger", "ANGSDcounts", "VCF" or "Chip". These values are not case-sensitive.
sampdepth.thresh	Minimum mean sample depth for retaining sample results. Default is 0.01.
snpdepth.thresh	Minimum mean SNP depth for retaining SNPs. Default is 0.01.
maf.thresh	Minimum MAF threshold for retaining SNPs. Default is 1e-7. Set to 0 to disable SNP filtering on MAF.
hirel.thresh	Lower threshold for reporting highly related individuals, and upper threshold for displaying positive control pairs which don't seem sufficiently related. Default is 0.9.
triallelic.thresh	Upper threshold for the proportion of ignored reads for the third allele – SNPs with a higher proportion are removed (as a triallelic variant). Relevant only to the ANGSDcounts input format. Default is 0.005 (0.5%).
cex.pointsize	Relative value of pointsize used in output graphics. This has a default value of 1.
functions.only	Set to TRUE to source GBS-Chip-Gmatrix.R for setting up functions (not reading data etc). Default is FALSE.
alleles.keep	Set to TRUE to retain an updated version of alleles. This object is needed for some downstream uses, e.g. for writing VCF files or for the calculation of linkage disequilibrium (using GUS-LD). Default is FALSE.
outlevel	Integer (1-9) determining the level of output created – higher numbers give more output. At present only two levels are active; 5 to 9 give the full output while 1 to 4 gives less output. A value less than 8 will supress the sampled alleles setup and analysis (reducing time). Default is 9 (all available output)
use.Rcpp	Set to FALSE to prevent the C++ versions of functions being used. Default is TRUE.
nThreads	The number of OpenMP threads to be used by C++. The default is 4. Using 0 means all available threads would be used.
iemm.thresh	Identity excess mismatch rate threshold for displaying non-matching results putatively from the same individual. The default is 0.05. Currently only used in the posC-EMM plot.
negC	character string containing a regular expression to be matched to *seq*ID to identify negative controls. The default is an empty string, in which case no checking is done. See below for more details.
negCsettings	a list of qualifiers to be passed to R's *grep* function when pattern matching for *negC* in *seqID*. The default is an empty list. See below for more details.
QQprobpts	numeric vector of probability points for plotting reference lines on QQ plots. The default is c(0.5,0.8,0.9,0.95,0.99). Set to numeric(0) to remove the lines. A new setting will be used in any subsetquent GBSsummary() call.
nogenos	set to TRUE to only get file information (sample and SNP information) from the input file. The default is FALSE. Currently information is retrieved for TagDigger, Tassel and uneak formats only.

Relatedness estimation program (GBS-Chip-Gmatrix.R)

This program performs some QC diagnostics, rudimentary data cleaning and defining a function (*calcG*) for relatedness estimation and reporting. A number of other functions are defined, such as those for checking and report on positive controls. Any procedures or output relating to depth are not implemented for chip data. The use of depth information to construct the GRM can be modified (see depth2K section).

If there are negative controls specified and found, they are summarised (normal output), reported (in file negCStats.csv and object *negCstats*) and removed from the data before further analysis. The negative controls are defined by matching the text *negC* in *seqID*. For example, if *negC* is

"NEG" then any seqID containing the string "NEG" is treated as a negative control. The *grep* function with usual default is used for pattern matching so that *negC* is used as a regular expression. This can be modified by setting *grep* arguments in *negCsettings*. For example,

```
negCsettings <- list(fixed=TRUE)
```

will match *negC* as is to *seqID*. i.e. does not treat it as a regular expression. Negative control checks are not invoked for chip data.

Samples with very low depth are dropped from the analyses. The threshold is a mean depth of *sampdepth.thresh* (default of 0.01, but can be set in the calling program) or with a maximum depth of one (including those with no genotype calls). Samples that are dropped are reported in the program output and are available in *seqID.removed*. The remaining number of samples is also reported in the output.

SNPs with no data or with a MAF (minor allele frequency) of zero are dropped. The remaining number of SNPs is reported. Sample and SNP removal is iterated until no more SNPs are removed.

Some basic statistics are reported: Proportion of missing genotypes is the number of SNP x individual combinations with no allele calls; Mean sample depth is the average depth (number of reads of either allele) for a sample.

The default action when sourcing GBS-Chip-Gmatrix.R is to read the data file and run some QC procedures, as well as define various functions. If functions.only is set to TRUE, then only the function definition occurs. The default action can then be mimicked using the pair of commands:

```
readGBS()
GBSsummary()
```

These functions are not yet described in the documentation. Additional processing can be inserted between these statements, for example to manually remove samples or SNPs. The following objects need to be maintained correctly, before *GBSsummary* is run: *nsnps, SNP_Names, seqID, nind, alleles*. If *GBSsummary* has been run once, it could be re-run, e.g. after merging results from the same individual. In that case (detected by the presence of *depth*), processing that uses *alleles* (which is not recalculated in *mergeSamples* unless *keep.alleles* is set to TRUE) is omitted. If *alleles* is present and corresponds to the current data, then *depth* could be removed so that it gets recalculated. This is mainly to obtain *genon* and *depth* which will be assumed to be present and correct, but it should be noted that *p* is not recalculated. *p* should remain unchanged when samples are merged, but could change, for example if the *sampdepth.thresh* is changed between calls to *GBSsummary*.

Some functions have been coded in C++ to improve speed. These will be used instead of the corresponding R functions if the libraries Rcpp and RcppArmadillo are installed, and use.Rcpp is TRUE (the default value).

Output - files

negCStats.csv contains call rates, along with mean sample depths for each sample identified as a negative control.

SampleStatsRaw.csv contains mean sample depths (for GBS data) for each sample before any SNPs or samples have been filtered out (by sampdepth.thresh, maximum sample depth <1, snpdepth.thresh, MAF=0).

SampleStats.csv contains call rates for each sample, along with mean sample depths (for GBS data).

AlleleFreq.png is a plot of allele frequencies calculated using different methods (and as given, if the uneak format is used).

CallRate.png shows a histogram of sample call rates (proportion of SNPs with a result for a sample).

SampDepth.png plots mean sample depth against median sample depth.

SampDepth-scored.png plots mean sample depth, over SNPs that are scored for the individual, against mean sample depth over all SNPs for the individual.

SampDepthHist.png is a histogram of mean sample depths

SampDepthCR.png plots mean sample depth against call rate.

SNPDepthHist.png is a histogram of SNP depths (number of reads of either allele averaged over samples)

SNPCallRate.png is a histogram of SNP call rates (proportion of samples with a result for a SNP)

SNPDepth.png plots SNP depth against mean SNP depth (on a log scale). This may reveal SNPs that are called infrequently, but when they are called have good depth (these SNPs may be near the boundary of a size selection step in the laboratory).

finplot.png plots Hardy-Weinberg disequilibrium (HWD) against MAF, shaded by the SNP depth. HWD is the proportion of (reference allele) homozygotes minus the expected proportion (under Hardy-Weinberg equilibrium). HWD is the same whichever allele is used in the calculation. The 'fin plot' may reveal sets of SNPs that do not follow Mendelian inheritance, for example apparent SNPs in duplicated regions.

HWdisMAFsig.png is similar to the fin pot, but with shading by *l10pstar*, the $\log_{10}$ p-value corresponding to the depth-adjusted chi-squared test statistic of Hardy Weinberg equilibrium (versions prior to v0.702 used the likelihood ratio test statistic for HWD).

LRT-QQ.png is a QQ plot for the likelihood ratio test statistic for HWD. Grey lines connect the x and y axis values corresponding to the cumulative proportions given in *QQprobpts*.

LRT-hist.png is a histogram of the likelihood ratio test statistic for HWD.

X2star-QQ.png is a QQ plot for the depth-adjusted chi-square test statistic for HWD. To allow comparison with values shown in HWdisMAFsig.png, *l10pstar* ≈ 0.77 + 0.218 * *x2star*. Grey lines connect the x and y axis values corresponding to the cumulative proportions given in *QQprobpts*. The right-hand side vertical axis shows the $-\log_{10}$ probabilities for the depth-adjusted chi-square test statistic (*x2star)* (the variable used for colour shading in *HWdisMAFsig.png*).

MAF.png is a histogram of the MAFs for each SNP (based on observed genotypes).

Variables defined

Variables that are set (in the R global environment) during the *readGBS* and *GBSsummary* execution include those shown in Table 2.

Table 2 Variables that are set in data reading, summarising and QC.

Variable	Description
nind	Number of samples analysed (after initial QC)
nsnps	Number of SNPs analysed (after initial QC)
seqID	Identifiers for each sample
seqID.removed	Identifiers for samples removed during initial QC.
SNP_Names	Identifiers for each SNP
chrom	chromosome label (character), if *gform* is Tassel
pos	chromosome position (numeric), if *gform* is Tassel

alleles	matrix (*nind* x 2**nsnps*) of read counts. The results for each SNP are in consecutive columns.
refalleles	a vector of reference allele bases (A, C, G or T) of length *nsnps*, when *gform* is TagDigger or VCF
altalleles	a vector of alternate allele bases (A, C, G or T) of length *nsnps*, when *gform* is TagDigger or VCF
genon	matrix (*nind* x *nsnps*) of numeric genotype calls 0 (homozygous alternate allele), 1 (heterozygous), 2 (homozygous reference allele), NA for missing
depth	matrix (*nind* x *nsnps*) of counts for each sample and SNP
samples	matrix (*nind* x *nsnps*) equivalent to *genon* but using a single read for each element of the matrix, with the read sampled all the reads for that genotype. This may be useful for diagnostics.
sampdepth	mean depth for each sample
snpdepth	mean depth for each SNP
callrate	mean call rate for each sample
SNPcallrate	mean call rate for each SNP
p	reference allele frequencies on the basis of allele counts
pg	reference allele frequencies on the basis of genotype calls
HWdis	Hardy-Weinberg disequilibrium (raw)
x2star	Depth-adjusted chi-squared test statistic of Hardy Weinberg equilibrium
l10pstar	$\log_{10}$ p-value corresponding to *x2star*
negCstats	data frame containing information (call rate and mean depth) on the negative controls

Reading data files

Function to read input file (readGBS)

This function reads the input file with the format specified in *gform*. The function is called by the main program (if *functions.only* is FALSE, the default) with the default settgins, but can be called by the user (normally with *functions.only* set to TRUE).

Usage: *readGBS*(genofilefn = genofile, usedt=”recommended”)
Arguments:

genofilefn	the name of the file to read. Defaults to *genofile*.
usedt	determines the use of the data.table package (if available). Set to “always” to force the use of data.table for Tassel input format. There may be problems with this setting with very large input files (> ~2billion elements). The default is “recommended”. It is planned to develop this feature for other input formats in the future. There may not be much (or any) advantage of using data.table with some formats because of the manipulations required after reading the file.

Value: NULL

Function to read TagDigger format files (readTD)

This function is for reading TagDigger files. It can be used by the main program (if *functions.only* is FALSE, the default), but can also be used to read additional files (e.g. to compare results in two different files). The variables *nsnps, SNP_Names, seqID, nind, alleles, refalleles* and *altalleles* are defined. See the section on the TagDigger format for more information.

Usage: *readTD*(genofilefn0 = genofile, skipcols=0)
Arguments:

genofilefn0	the name of the file to read. Defaults to *genofile*.
skipcols	the number of columns of input to ignore. Defaults to 0.

Value: NULL

Function to read variant call format (VCF) files (read.vcf)

This function is for reading VCF files with AD and/or GT fields. It can be used in place of *readGBS*. The AD field is used preferentially. If only the GT field is available, the genotype call is taken as having infinite depth in which case the data are analysed in the same way as “chip” data. Allele frequencies are calculated from the allele counts if the AD field is present, otherwise from genotypes. The function will fail if there are more than .Machine$integer.max genotypes except for the special case where ethe vcf file has only GT data. The variables *nsnps, SNP_Names, seqID, nind, chrom, pos, genon, depth, p, alleles, refalleles* and *altalleles* are defined. Currently the function requires the package data.table, and, if the VCF file is gzipped (.gz), the package R.utils (depending on the version of data.table).

Usage: *read.vcf*(vcffile=genofile)
Arguments:

vcffile	the name of the VCF file to read. Defaults to *genofile*.

Value: NULL

Function to read SNP chip (genotype) files (readChip)

This function is used by *readGBS* for reading files of genotype data, when *gform* is “Chip”. However, it can be used directly to access non-default parameter settings (sep and row.names). The variables *nsnps, SNP_Names, seqID, nind, genon, depth,* and *p* are defined. Currently the function does not use the package data.table.

Usage: *readChip*(genofilefn0 = genofile, sep=",", row.names=FALSE)
Arguments:

genofilefn0	the name of the file to read. Defaults to *genofile*.
sep	the separator used in the file, default is a comma
row.names	indicates whether the first column has row names and is not labelled (such as written with write.csv with default settings The default is FALSE.

Value: NULL

Data set-up and summaries (GBSsummary)

GBSsummary creates relevant objects (in the global environment) and produces summaries. See the documentation for ‘Relatedness estimation program’ for more details. *GBSsummary* is automatically run after *readGBS,* when sourcing the R scipt, if *functions.only* is FALSE.

Manipulating data

Function to create (a new version of) the samples object (redosamples)

This function creates *samples*, a matrix (*nind* x *nsnps*) equivalent to *genon* but using a single read for each element of the matrix, with the read sampled all the reads for that genotype. This may be useful for diagnostics. It is used for creating the G3 version of the relatedness matrix. It may be used for estimating model parameters used in the depth functions.

Usage: redosamples ()
The function had no arguments. It assumes that *alleles* corresponds to the data (set *alleles.keep* to TRUE if using *mergeSamples* prior to *redosamples*). A (new) version of *samples* is created. *redosamples* is run as part of *GBSsummary* if *outlevel* > 7.

Function to (re)create genon and depth objects (alleles2g)

This function creates *genon* and *depth* from *alleles.* It assumes that *alleles* corresponds to the data (*nind, nsnps, seqID* etc).
Usage: *alleles2g* (dogenon=TRUE, dodepth=TRUE)
Arguments:

dogenon	if TRUE (the default), creates *genon*
dodepth	if TRUE (the default), creates *depth*

Function to store the main data objects (parkGBS)

This function copies the main data objects (*nsnps, SNP_Names, seqID, nind, alleles, AFrq* if relevant) into a list for future reference. Normally used when multiple GBS datasets are being read.

Usage: *parkGBS*()
The function had no arguments. It assumes that *alleles* corresponds to the data (set *alleles.keep* to TRUE if using *mergeSamples* prior to *parkGBS*)

Function to activate a 'parked' workspace (activateGBS)

This function restores the objects in a list created by *parkGBS* back into the global environment.

Usage: *activateGBS*()

Function to combine two sets of GBS data (joinGBS)

This function combines 2 sets of GBS data. At least one of these sets is in a list (normally created with *parkGBS*). The number of samples output will be the sum of the numbers in the two input objects. These can be merged using *mergeSamples(seqID, …)* if required. If there are SNP_Names in common in the two input objects they are assumed to be the same SNPs, unless *uniqueSNPs* is TRUE. The function only uses and outputs the main data objects (those that are stored by *parkGBS)*.

Usage: *joinGBS(join1, join2=NULL, replace=TRUE, uniqueSNPs=FALSE)*
Arguments:

join1	the name of a list containing the data for the first dataset (normally created with *parkGBS*).
join2	the name of a list containing the data for the second dataset (normally created with *parkGBS*), or NULL (the default) in which case the required objects are retrieved from global environment.
replace	If TRUE (the default), the corresponding objects in the global environment will be replaced (or removed if not available) by those from the combined data.
uniqueSNPs	if TRUE, the SNPs are taken to be different in the two input data sets. If some names are in common, the ones in the first dataset are renamed by appending ".1" to the names. The default is FALSE.

Functions for merging results for the same individual (mergeSamples, mergeSamples2)

A function, *mergeSamples*, for merging samples (adding allele counts across results) from the same individual. The function *mergeSamples2* is similar, see below. The function *posCreport* can be used beforehand to check if the results to be merged appear to be from the same individual.

Usage *mergeSamples* (mergeIDs, indsubset, replace=FALSE, IDouttype=1, rowsumfn=rowsum2)
Arguments:

mergeIDs	a vector of identifiers for all *nind* samples, such that samples that have the same identifier are to be merged.
indsubset	a vector of integers (between 1 and *nind,* inclusive) of samples to be retained. The default is to use all all samples.
replace	if TRUE, place the relevant objects (all or some of *nind, seqID, genon, depth, alleles, sampdepth, snpdepth, pg, callrate, SNPcallrate*) in the global environment instead of the output object. The default is FALSE.
IDouttype	determines how the IDs of merged samples are formed. (Under development). See the description of seqID in the output below.
rowsumfn	a function that works in the same way as rowsum (only needs to work for matrices). The default is *rowsum2*, a supplied function that partitions the data to circumvent a rowsum error when the output has more than the maximum integer allowed (currently 2^{31}-1) elements.

Value: a list of the following objects (if relevant):

mergeIDs	a vector of identifiers, as per the input, but ordered as in the other output objects (and with unique values)
nind	the length of *mergeIDs*
seqID	normally one of the seqIDs that correspond to the *mergeIDs*. If *IDouttype* is 1 and the seqIDs can be broken into five parts, using an underscore (_) as a separator, then the second part will be returned if unique, otherwise replaced by "X", the third part by the number of results merged, the fourth part by "0" and the fifth part will be replaced by "merged". If *IDouttype* is 2 and the seqIDs can be broken into parts, using an underscore (_) as a separator, then the first non-unique part is replaced by "merged" and others by "0". If *IDouttype* is 3 and the seqIDs can be broken into parts, using an underscore (_) as a separator, then unique parts (after the first part) are returned, non-unique character parts are replaced by "X" and non-unique character parts are replaced by 0.
genon	genotype (0/1/2) matrix after merging, if *genon* exists before merging.
depth	depth matrix after merging, if *genon* exists before merging.
alleles	alleles matrix after merging, if *alleles.keep* is TRUE or, if *genon* does not exist before merging
sampdepth	sample mean read depths after merging, if *genon* exists before merging.
snpdepth	SNP mean read depths after merging, if *genon* exists before merging.
pg	allele frequencies based on genotype calls, after merging.
nmerged	number of results merged (1, if not merged) for each individual.
callrate	sample call rates
SNPcallrate	SNP call rates

Normally these objects would be used to replace their corresponding values before the merge, but this is not done automatically unless *replace* is set to TRUE. When *replace* is TRUE, the output object contains only *mergeIDs* and *nmerged*. Note that some objects are not merged (e.g. the allele depth matrix, *alleles,* if *alleles.keep* is FALSE) and that the diagnostics produced when sourcing GBS-Chip-Gmatrix.R are not re-done by this function. If *mergeSamples* is run before *GBSsummary* (when *functions.only* is TRUE, and after the data is read), then it produces a reduced set of outputs, corresponding to those existing before the merge. In particular, it will have an *alleles* object, but neither *genon* nor *depth*.

mergeSamples with *rowsumfn* = *rowsum* will fail when the number of elements for a merged object (*genon*, and *alleles* if *alleles.keep* is TRUE) exceeds the maximum integer allowed (currently 2^{31}-1). Using *rowsumfn* = *rowsum2* will circumvent this issue. *mergeSamples2* provides an alternative strategy to allow larger merges when some of the records do not require merging, with the limit being 2^{31}-1 elements in the subset of *genon* containing *mergeIDs* with at least two observations in the input data. *mergeSamples2* may be faster than *mergeSamples* when most of the data is not being merged. *mergeSamples2* does not contain some of the other functionality of *mergeSamples* (yet).

Function to remove samples from objects (samp.remove)

This function removes samples from the relevant objects (*alleles, depth, sampdepth, seqID nind*). It would normally be used between calls to *readGBS* and *GBSsummary*.

Usage: *samp.remove*(samppos = NULL, keep=FALSE)
Arguments:

samppos	the positions of the samples to remove. Defaults to NULL.
keep	If TRUE, the samples with positions samppos will be kept and other samples removed. Default value is FALSE

Function to remove SNPs from objects (snp.remove)

This function removes SNPs from the relevant objects (*p, nsnps, SNP_Names, alleles, depth,* and some others). It would normally be used between calls to *readGBS* and *GBSsummary*.

Usage: *snp.remove*(snppos = NULL, keep=FALSE)
Arguments:

snppos	the positions of the SNPs to remove. Defaults to NULL.
keep	If TRUE, the SNPs with positions snppos will be kept and other SNPs removed. Default value is FALSE

Depth functions and allele sampling modelling

The GBS-Chip-Gmatrix.R program defines a default function for calculating "*K* values", as well as alternate functions (using alternate allele sampling models) and a function to reset the default to one of the alternatives. These functions are relevant for used both self-relatedness estimation and pedigree assignment diagnostics. If a different depth model is required for undertaking depth-adjusted calculations (self-relatedness, mismatch statistics etc) this *depth2K* function should be re-defined before using the relevant function. Note that *depth2K* is defined or re-defined in the global environment and should not be defined in any user functions. K is the probability of observing an AA genotype, given that the true genotype is AB and the read depth is *k*. These models will be discussed in more detail elsewhere. The function is used within *calcG* for calculating the self-relatedness for G5, in the pedigree assignment program, for calculating expected mismatch rates, in the function for calculating identity mismatch rates as well as for population genetic statistics such as Hardy-Weinberg tests or Fst calculations.

Some utility functions are available for estimating parameters for depth models.

Convert read depths to K values (depth2K)

A function *depth2K* is defined. This function takes a vector of read depths and returns the corresponding set of *K* values. Initially the function is defined using a binomial sampling model (the number of A alleles is binomial with probability parameter 0.5 and sample size the read depth).

Convert read depths to K values using a beta-binomial model (depth2Kbb)

depth2Kbb is an alternate depth function which uses a beta-binomial model. The beta-binomial distribution has two parameters, α and β, but here these are set to be equal, so that P(AA|AB, *k*=1) = 0.5.

Usage: *depth2Kbb* (depthvals, alph=Inf)
Arguments:

depthvals	a vector of read depths
alph	the value of α (and also β) – the default is to use Inf, in which case the binomial model is used.

Convert read depths to K values using a modified-p model (depth2Kmodp)

depth2Kmodp is an alternate depth function which uses a 'modified-p' value for 2nd and subsequent reads. The modified-p can be thought of as the probability of seeing the same allele as in the previous read (for that SNP) for a true AB genotype, although because we are only interested in the probability of all reads being the same allele, it is also the probability of seeing the same allele as *all* previous reads (for a true AB genotype).

Usage: *depth2Kmodp* (depthvals, modp=0.5)
Arguments:

depthvals	a vector of read depths
modp	the modified probability – the default is 0.5, which gives the binomial model. Normally a value ≥ 0.5 would be used to reflect an increased chance of seeing the same allele as in the previous read.

Function to re-define depth2K(depth2Kchoose)

depth2Kchoose is a function to re-define *depth2K* to one of the alternative models.

Usage: *depth2K* <- *depth2Kchoose* (dmodel="bb", param)
Arguments:

dmodel	the model to use, either "modp" (to use *depth2Kmodp*), or "bb" to use *depth2Kbb* – the default is "bb" (also used if any other string is used)
param	the parameter to use for the alternative function, used for alph for the bb model, and modp for the modp model.

Calculate sum of squared Inbreeding deviations (ssdInb)

A function to use for determining the fit of an allele sampling model on the basis of its inbreeding estimates compared to a set of given inbreeding values.

Usage: ssdInb (dpar=Inf, dmodel="bb", Inbtarget, snpsubset, puse, indsubset, quiet=FALSE, quieti=TRUE)

Arguments:

dmodel	the depth model (allele sampling model, see the section on depth functions) to be used. This must be one of "mp" (modified p) or "bb" (beta-binomial). The default is to use the beta-binomial model.
dpar	the parameter for the depth model
Inbtarget	a set of inbreeding values to compare against. If *ssdInb* is being used to estimate dpar, it can be achieved by minimising the sum of squared deviations from *Inbtarget*.
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of the SNPs to use in the calculation. The default is to use all SNPs.
puse	a vector or matrix of (reference) allele frequencies to use in the calculations. See the *puse* arguments for *calcGdiag* and *calcG* for further details.
indsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals for which relatedness matrices will be calculated. The default is to calculate for all individuals. The lengths of *indsubset* and *Inbtarget* should be the same. There is currently no check for this.
quiet	Set to TRUE to prevent parameter and EMM sum of squares being displayed. The default is FALSE.
quieti	Set to TRUE to prevent reporting back from the inbreeding calculation process (*calcGdiag*). The default is TRUE.

Details: Returns the sum of squared deviations of inbreeding calculated with the specified depth model from the supplied inbreeding values. The function can be used to estimate the depth model parameter, e.g. with optimise(ssdInb,lower=0,upper=20, tol=0.001, Inbtarget=<vector of inbreeding>).

Calculate sum of squared identity excess mismatches (ssIEMM)

A function to use for determining the fit of an allele sampling model on the basis of identity excess mismatches (IEMM) for pairs of results for a set of individuals.

Usage: ssIEMM (dpar=Inf, dmodel="bb", uind1, uind2, snpsubset, puse, inbreed.method="ignore", quiet=FALSE)

Arguments:

dmodel	the depth model (allele sampling model, see the section on depth functions) to be used. This must be one of "mp" (modified p) or "bb" (beta-binomial). The default is to use the beta-binomial model.
dpar	the parameter for the depth model
uind1	the positions of the first result for each individual (e.g., in seqID)
uind2	the positions of the second result for each individual (e.g., in seqID)
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of the SNPs to use in the calculation. The default is to use all SNPs.
puse	a vector or matrix of (reference) allele frequencies to use in the calculations. See the *puse* arguments for *calcGdiag* and *calcG* for further details.
inbreed.method	determines what value of inbreeding to use in *mismatch.ident*. Using "mean" will use the mean inbreeding of the pair of results being compared. Using "ignore" (the default) will use an inbreeding value of 0.

quiet	Set to TRUE to prevent parameter and EMM sum of squares being displayed. The default is FALSE.

Details: Returns the sum of squares if IEMM values calculated with the specified depth model. The function can be used to estimate the depth model parameter, e.g. with optimise(ssIEMM,lower=0,upper=20, tol=0.001, uind1=<vector of first indices>, uind1=<vector of first indices>, inbreed.method="mean"). An data.frame *mmstats.last* with columns *mmrate* and *exp.mmrate* is placed in the global environment. In particular, this has the value of these statistics for the last iteration if *ssIEMM* is passed to *optimise*.

Function for initialising and/or retrieving beta-binomial parameters (bbinit)

When using an alternative allele sampling model, it may be appropriate to apply different parameters to different subsets of the data. The *bbinit* function helps set up and maintain a data frame of group IDs and beta-binomial alpha parameters.

Usage bbinit(bbgroups, bbfile=NULL)

Arguments:

bbgroups	a vector of group identifiers. Only the unique values of the vector are used, so it may be given as the identifiers for each sample or as the unique set (of interest).
bbfile	The name of a .csv file containing a previously saved set of beta-binomial alpha values for some or all of the groups. The default is NULL in which case no file is read. The first column should contain the grouping identifiers and the second column should contain alpha values. Further columns may be included and will be retained.

Details: returns a data frame of unique *bbgroups* values (including any stored in *bbbfile*) in the column *bbgroup* and their alpha values in the column *bbalpha*. Alpha values are from *bbfile* if the group is in that file, or set to Inf otherwise.

Function for estimating beta-binomial parameters (bbestimate)

Estimates beta-binomial alpha values by minimising the sums of squared differences between inbreeding calculated with a beta binomial model and a given set of inbreeding values.

Usage bbestimate <- function(mergeIDs, bbgroups, bbinfo, Inbstd, repid, snpsubset, puse=p, mindepth.bb=0.1, preoptim=TRUE, groupname="group", repname="rep")

Arguments:

mergeIDs	a vector of identifiers for all *nind* samples, such that samples that have the same identifier are from the same genetic identity (individual).
bbgroups	a vector of group identifiers of length *nind*. These are to be matched to *bbinfo$bbgroup*.
bbinfo	a data frame containing *bbgroup* and *bbalpha*. When *bbalpha* is Inf, the alpha parameter will be estimated for that group. If there are subsets where alpha estimation is not wanted, those subsets should have a *bbgroups* value that does not appear in *bbinfo$bbgroup*.
inbstd	A set of inbreeding values to use as the target set
repid	a vector of 'replicate' IDs, where there is a single result for each mergeIDs × repid and results with the same repid represent a logical set of results, e.g. sequenced together.
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of the SNPs to use in the calculation. The default is to use all SNPs.
puse	a vector or matrix of (reference) allele frequencies to use in the calculations. See the *puse* arguments for *calcGdiag* and *calcG* for further details.
mindepth.bb	the minimum depth for a result to be used in the alpha estimation process. The default is 0.1.
preoptim	if FALSE, the optimisation uses a lower bound of 0 and an upper bound of 200; if TRUE (the default) an initial guess for alpha is made (based on mean differences in inbreeding estimates and considering mean depth) and the search bounds are set at half to double this value.

groupname	a name for referring to bbgroup, the default is "group".
repname	a name for referring to the replicate factor for the same values of *mergeID.* The default is "rep".

Details: returns an updated version of *bbinfo* with alpha estimates for cases where the alpha value was previously Inf. For any value of *bbinfo$bbgroup,* if the mean of *Inbstd* is greater than the mean estimated inbreeding for that group when using the binomial model, the alpha estimation is stopped and an arbitrary high value (1,000,000) of alpha is given (the beta-binomal model with this value will be essentially the binomial model). If *bbinfo* was created using a file of previous values (in *bbinit*) then the column names of the output will match those from the imported file.

InbCompare-<bbgroup>.png a scatterplot matrix, for each *bbgroup* where alpha has been estimated, containing inbreeding estimated different ways: 1) merging results on *mergeIDs* and using the binomial model, 2) merging results on *mergeIDs* and using the beta-binomial model with the estimated alpha, 3) *Inbstd* (assumed to use a single read per rep, and labelled as such), and if there are exactly 2 levels of the replicate factor, using the relatedness estimates (which don't depend on the depth function) between (unmerged) replicates for each *mergeIDs*.

Convert depths to effective depths using beta-binomial alpha values (bbeffdepth)

This function converts depths to 'effective depths' based on a set of beta-binomial alpha values for each sample. The effective depth with the binomial *depth2K* function will return the same *K* values as the beta-binomial *depth2K* function using the original depths. This allows analyses where different alpha values can be used for different samples: the depths are replaced by the effective depths and the binomial *depth2K* function is used.

Usage bbeffdepth(bbgroups, bbinfo)

Arguments:

bbgroups	a vector of group identifiers of length *nind*. These are to be matched to *bbinfo$bbgroup*.
bbinfo	a data frame with first column containing beta binomial groups and 2nd column containing their alpha values. If there are subsets where the binomial allele sampling model should be used, those subsets should have a group value that does not appear in the first column.

Details: returns a depth matrix. This matrix will have non-integer values, but can still be used with other KGD functions.

Function for calculating identity mismatch rates (*mismatch.ident*)

Mismatch rates for comparing two results from putatively the same individual. Currently under development. Used by *posCreport*.

Usage mismatch.ident(seqID1, seqID2, snpsubset=1:nsnps, puse=p, mindepth.mm=1, inbreed=0)

Arguments:

seqID1	seqID for first result
seqID2	seqID for second result
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of SNPs to be compared. The default is to use all SNPs.
puse	a length *nsnps* vector of allele frequencies to use in the calculations. The default is to use allele frequencies calculated on the basis of allele counts.
mindepth.mm	the minimum depth for a genotype to be used in the comparison.
inbreed	the inbreeding of the individual. The default is 0. A sensible alternative is to use the mean of the inbreeding estimated from each of the genotyping results.

Value: a list containing mmrate (raw mismatch rate), ncompare (number of SNPs compared), exp.mmrate (expected mismatch rate) and a data frame snpmmstats with variables mm (was there a mismatch) and expmm (expected snp mismatch rate).

Function for reporting on negative controls (*negCreport*)

A function, *negCreport*, for reporting on samples which are supposedly blank (negative controls). This function is used in *GBSsummary* if there are results corresponding to the *negC* setting, but can also be used manually, for example if negative controls cannot be identified by their ids.

Usage negCreport(negpos, removeneg=FALSE)

Arguments:

negpos	a vector of positions corresponding to the seqID order, of results considered to be from negative controls.
removeneg	if TRUE, the corresponding results will be removed from the analysis (regardless of their results). The default is FALSE, but in *GBSsummary* a value of TRUE is used (i.e., results identified by *negC* are removed by *GBSsummary*).

Value: a data.frame containing seqID, callrate and meandepth for the specified negative controls.

negCStats.csv contains call rates, along with mean sample depths for each sample identified as a negative control.

Function for reporting on positive controls (*posCreport*)

A function, *posCreport*, for reporting on samples which are supposedly from the same individual. These will normally be one or more positive controls but may also be repeat runs.

Usage posCreport(mergeIDs, Guse, sfx = "", indsubset, Gindsubset, snpsubset=1:nsnps, puse=p, snpreport=FALSE, inbreed.method="ignore", quiet = FALSE)

Arguments:

mergeIDs	a vector of identifiers, ordered as in *Guse*, where samples from the same individuals are given the same identifier
Guse	the G matrix for comparing samples
sfx	text to be included in output file names to allow output from multiple calls or runs to be identified
indsubset	a vector of integers (between 1 and *length(mergeIDs),* inclusive) of individuals in *mergeIDs* (and *Guse*) to be compared. The default is to use all individuals.
Gindsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals from the full data in *Guse* (normally the same as used for *indsubset* when calling *calcG* to obtain *Guse*)
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of SNPs to be compared for mismatch rates. The default is to use all SNPs. If an empty set (integer(0)) is given, comparisons using relatedness estimates only will be undertaken (this is useful if comparisons are to be made from saved relatedness estimates from different analyses).
puse	a length *nsnps* vector of allele frequencies to use in the mismatch calculations. The default is to use allele frequencies calculated on the basis of allele counts.
snpreport	if TRUE, information is returned on SNP mismatch rates, over all pairs of duplicate sample results. The default is FALSE.
inbreed.method	determines what value of inbreeding to use in *mismatch.ident*. Using “pairmean” (or text containing “pair”) will use the mean inbreeding of the pair of results being compared. Using “allmean” (or text containing “all”) will use the mean inbreeding of all results for a particular value of *mergeIDs*. Using “ignore” (the default) will use an inbreeding value of 0.
quiet	if TRUE, supress the display of pairs with self-relatedness statistics outside the thresholds. The default is FALSE.

Value: when snpreport is FALSE, a data frame containing columns mergeID (the ID given in *mergeIDs*), nresults (the number of runs with this ID), selfrel (the average self-relatedness), meanrel (the mean relatedness between all pairs with the given value of mergeID), minrel (the minimum relatedness between all pairs with the given value of mergeID), meandepth (mean of *sampdepth*), mindepth (minimum *sampdepth*), meanCR (mean call rate). Only values of mergeID

with nresults >1 are included. When snpreport is TRUE, a list is returned with elements posCstats containing the data frame just described and a data frame called snpstats with rows corresponding to SNPs (length *nsnps*) and columns snpcount (the number of times the SNP was compared), snpmm (the number of mismatches for that SNP), expmm (the expected mismatch rate) and mmrate (the actual mismatch rate).
Details: The function displays (when quiet is FALSE) pairs of results (relatedness estimate, mean depth for each sample and IEMM) where the estimated relatedness is less than 1 and below the selfrel by at least 1- *hirel.thresh*. When there are results generated by these criteria and there are more than two genotype results with the corresponding *mergeID*, those results are grouped into sets that appear to be consistent. Each result in a consistent set will have a scaled relatedness (referred to as srel in the output) with another result in the set of at least *hirel.thresh*, were the relatedness is scaled so the self-relatedness is 1. In addition, the following files are generated:

posCchecks<sfx>.txt a copy of the results displayed on the default output (i.e. low relatedness pairs)

posCreport<sfx>.csv contains the data frame that was returned by the function

SelfRel<sfx>.png a plot of *meanrel* against *selfrel*. The line of equality is shown in red. A grey line gives the boundary where relatedness is lower than 1 and lower than selfrel by more than 1 - *hirel.thresh* (as a guide for results to check).

posC-MM<sfx>.png a plot of mean (over pairs of the same individual) raw mismatch rate against mean expected mismatch rate. The line of equality is shown in red, while a grey line denotes when the difference (identity EMM, IEMM) is greater than the threshold *iemm.thresh*.

posC-EMM<sfx>.png a scatterplot of matrix mean of *selfrel, meanrel* and IEMM.

posC-SNPMM<sfx>.png a plot of mean (over all pairs of the same individuals) raw mismatch rate against mean expected mismatch rate for each SNP. The line of equality is shown in red. This plot is produced when snpreport is TRUE.

Plate plot function (*plateplot*)

The *plateplot* function can be used for quality control based on laboratory plate layouts. The user must supply this layout in *plateinfo*. If there are more than one observation for a given plate position, then the sum for those observations is used (but currently the function is intended to be used with a single value for each position). Positions without data will be displayed as white.

Usage: plateplot (plateinfo, plotvar=sampdepth, vardesc="", sfx="", neginfo, negcol="grey", focusinfo, focuscol="black", colpal = hcl.colors(80,palette="YlGnBu", rev=TRUE), vflip=FALSE)

Arguments:

plateinfo	a data frame with elements row, column and optionally subplate (anything else ignored). The rows in *plateinfo* should correspond to the elements in *seqID* (at time of use). row and column contain plate row and column labels for an observation. These can be text or numeric, but plate physical order and sort order of these label should match. If *plateinfo* has a variable *subplate* present, this should represent plate identifiers for an earlier step in the lab process than for the plate represented by the *row* and *column* positions. In this case an additional plot is produced distinguishing the *subplate* levels.
plotvar	a variable with values to be plotted (using a colour scale). The default is *sampdepth*.

vardesc	A descriptive name for *plotvar*, used to label the colour scale on the plot(s). A length zero string (the default) will result in the name of the variable given in *plotvar* to be used.
sfx	A suffix to use in output file names to identify which function call has produced that output.
neginfo	A data frame with elements row, column identifying the positions of any negative controls. These will be identified on the plots with two diagonal lines (X) through the position. If not given negative control plotting is ignored.
negcol	the colour of the lines identifying negative control positions. The default is "grey".
focusinfo	A data frame with elements row, column identifying the positions of any wells to be highlighted (e.g. positive controls). These will be identified on the plots with a border around the position. If not given highlighting is ignored.
focuscol	the colour of the lines to highlight the positions in *focusinfo*. The default is "black".
colpal	a colour palette to be used for displaying *plotvar*. The entire palette is used to cover the data range (the value corresponding to a given colour will change depending on the data). It is advised that the palette does not include white as this will be the colour for any positions without data (e.g., failed), which are useful to have identified. The default palette uses the "YlGnBu" (yellow – green – blue) palette (or similar, if not available). Other possibilities include viridis colours (yellow low, blue high, e.g., rev(hcl.colors(80),rev=TRUE), requires R.version ≥ 3.6) or heatmap colours (e.g. yellow to red: rev(heat.colors(80))[25:80])
vflip	if TRUE the vertical axis will be flipped (ordered from the top down)

Details: Colour plots of the variable in plate layout are produced as given below. The function does not return an object to R.

Plate<sfx>.png Colour plot (using colpal) of the variable in plate layout. A colour legend is included with the minimum, mean and maximum of the variable (summed, if relevant) indicated.

Subplate<sfx>.png Colour plot of the variable in plate layout with darker colours indicating higher values and different base colours used for each level of *subplate*. A colour legend is included with the minimum, mean and maximum of the variable indicated (summed, if relevant). This plot is produced only when *plateinfo* contains a *subplate* column.

Hardy-Weinberg statistics function (*HWpops*)

A function, *HWpops*, for calculating Hardy-Weinberg statistics, by population if provided, is defined.

Usage: HWpops(snpsubset, indsubset, populations=NULL, depthmat = depth)

Arguments:

snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of SNPs calculate the statistics for. The default is to use all SNPs.
indsubset	a vector of integers (between 1 and *nind,* inclusive) of individuals to use for the calculations. The default is to use all individuals.
populations	a vector of length *nind* containing population labels. The default is NULL in which case a single population of all individuals is used.
depthmat	matrix (*nind* x *nsnps*) of depth values. The default is depth which should normally be used.

Value: a list of the following objects:

HWdis	a matrix of Hardy-Weinberg disequilibrium values, populations in rows, SNPs in columns, population names used for row names.

l10LRT	a matrix of -$\log_{10}$ p-values from the likelihood ratio test assuming observed genotypes are true, populations in rows, SNPs in columns, population names used for row names.
x2star	a matrix of x2star values (depth-adjusted chi-squared test statistic of Hardy Weinberg equilibrium), populations in rows, SNPs in columns, population names used for row names.
l10pstar	a matrix of -$\log_{10}$ p-values corresponding to *x2star*, populations in rows, SNPs in columns, population names used for row names.
maf	a matrix of minor allele frequencies (based on observed genotypes), populations in rows, SNPs in columns, population names used for row names.
l10pstar.pop	(if there is more than one population) a vector of -$\log_{10}$ p-values for each SNP corresponding to a combined within populations test of *x2star*. Non-missing *x2star* values are summed over populations and the p-value calculated from the chi-squared distribution with degrees of freedom equal to the number of *x2star* values summed.

Allele frequency function (*calcp*)

A function, *calcp*, for calculating allele frequencies (for all SNPs) of the reference alleles, is defined.

Usage: calcp(indsubset, pmethod="A")

Arguments:

indsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals for which are to be used for allele frequency estimation. The default is to use all individuals.
pmethod	a method for calculating the frequencies, being one of "A" (calculate on the basis of allele counts) or "G" (calculate on the basis of genotype calls). The default is "G" for chip data (at least one depth is infinite) and "A" for other data.

Value: a vector of allele frequencies

Warning when using this after *mergeSamples*: pmethod A uses the object *alleles*, which is not recreated during the merge unless alleles.keep is set to TRUE (not the default setting), so indsubset refers to sample positions prior to the merge. pmethod G uses *genon* whose positions are those following the merge.

If pmethod A is used and any of the allele counts are Inf (e.g. include chip data) then allele counts are truncated at 2 and halved, if both alleles have a count > 0, before the calculations, to enable a value to be calculated. This may or may not be sensible depending on the situation (consider using pmethod G instead – this will be enforced if *genon* is available).

Genomic relatedness function (*calcG*)

A function, *calcG*, for calculating the genomic relatedness, is defined. This may be invoked several times with different options.

Usage: calcG(snpsubset, sfx="", puse, indsubset, depth.min=0, depth.max=Inf, npc=0, calclevel=9, cocall.thresh=0, mdsplot=FALSE, mindepth.idr = 0.1, withPlotly=FALSE, withHeatmaply=withPlotly, plotly.group=NULL, plotly.group2=NULL, samp.info=NULL, samptype="diploid", use.irlba = FALSE, cocallout=FALSE)

Arguments:

snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of the SNPs to use in the calculation. The default is to use all SNPs.
sfx	A suffix to use in output file names to identify which function call has produced that output.
puse	a vector or matrix of (reference) allele frequencies to use in the calculations. The default is to use allele frequencies calculated on the basis of allele counts. The values (for the snps in *snpsubset*) should be greater than 0 and less than 1. For the vector form, either a vector of the same length as

	snpsubset or one of length *nsnps* (in which case the function subsets these using *snpsubset*) can be supplied. If a matrix is supplied, it is the taken as the matrix of allele frequencies for individuals x snps. The number of rows can be the length of *indsubset* or *nind*, while the number of columns can be the length of *snpsubset* or *nsnps*, and if needed the function subsets to the required set.
indsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals for which relatedness matrices will be calculated. The default is to calculate for all individuals.
depth.min	The minimum depth for a SNP result for an individual to be used.
depth.max	The maximum depth for a SNP result for an individual to be used.
npc	The number of principal components of the 'G5' relatedness matrix (Gpool for *samptype* = "pooled") to display. If *npc*≤0, then the heatmap plot is omitted, but otherwise |*npc*| is used for *npc*. If *npc*=0 (the default) the principal component analysis is omitted.
calclevel	specifies the amount of calculation and output produced: 1 gives G5 (see below) and intermediate results only, 2 gives G5 and reports using G5, 3 gives all types of G available and 9 gives these and all reporting available.
cocall.thresh	Samples may be removed so that co-call rates (the proportion of SNPs with a call in both of a pair of samples) for heatmap and PCA analyses are above this value. Firstly, if *cocall.thresh* ≥0, samples with a maximum SNP depth of 1 are removed. The further samples are removed successively, with the sample appearing the most often in pairs not meeting the criterion removed at each step, until all pairs meet the criterion. The removal of these samples under the default threshold allows the heatmap and PCA analyse to be performed (no NAs in the relationship matrix used).
mdsplot	if TRUE and the conditions for plotting the principal components are met, a two-dimensional multidimensional scaling plot (principal coordinates plot) is also produced. The default is FALSE.
mindepth.idr	minimum depth for including samples in the self-relatedness (or inbreeding) regression on log(sample depth), applied after any filtering specified in the other call parameters.
withPlotly	If TRUE, then plotly graphs are produced, else if FALSE the standard plots are produced. The default is FALSE.
withHeatmaply	If TRUE, then plotly graphs are produced for the heatmap, else if FALSE the standard plots are produced. The default is the *withPlotly* setting.
plotly.group	A character vector of length equal to the number of individuals (in *indsubset*). Gives grouping on the plotly graphs in terms of different coloured points. The default is NULL (no colouring).
plotly.group2	A character vector of length equal to the number of individuals (in *indsubset*).. Gives grouping on the plotly in terms of different points. The default is NULL (no grouping with symbols).
samp.info	A list where each element is a character vector of length equal to the number of individuals (in *indsubset*). Used to provide "hover" information for plotly graphs. The default is NULL, in which case the seqID is used.
samptype	the type of samples being analysed. Currently only two values are allowed: "diploid" (the default) and "pooled" for the case where pools of individuals are sequenced. Any other value is treated as "diploid".
use.irlba	if this is TRUE and the irlba package is available, an alternative approximate method (known as IRLBA) is used for principal components analysis. The default is FALSE.
cocallout	if this is TRUE, a matrix *cocall* is included in the output. This contains the cocall counts for for each pair of individuals (call counts on the diagonal).

<u>Value</u>: a list of relatedness structures: G1, G4d (diagonal elements of G4), G5, Gpool, indsubset (as supplied to the function), nsnps, the number of SNPs used (length of snpsubset), samp.removed (positions of samples removed to ensure the cocall.thresh criterion), PC, the output

of the principal components analysis (if |*npc*|>0) and *cocall* (if *cocallout* is TRUE), the matrix of cocall counts.

The G*n* relatedness matrices are described in Dodds *et al*. (2015), except that a range of allele sampling models can be incorporated for the diagonal of G5 – see the depth2K section below. Also, non-integer depths can be used (only relevant for diag(G5)) and depths ≥ 1.001 are used for calculating diag(G5). This allows "effective" depths to be used (e.g. as a way of combining data where different allele sampling models have been used in different subsets).

Gpool is NULL except when *samptype* is "pooled" in which case Gpool is another relatedness matrix calculated in the same way as G5 (including the diagonal adjustment for depth), except that 2 x the proportion of reference alleles is used as the numeric genotype (rather than the number of reference alleles in the observed genotype). This is similar to the approach of Reverter *et al*. (2016) for pooled SNP-chip data, and Cericola *et al*. (2018) for pooled GBS data, except that no adjustment is made for pool size as in those studies – this is left to the user (e.g. Gpool could be multiplied by the pool size as in Cericola *et al*. (2018) or scaled so that diagonal elements are 1 as in Reverter *et al*. (2016)). Note that the diagonal adjustment for depth has not been theoretically verified, and therefore Gpool should be considered as under development. When *samptype* = "pooled" the outputs are based on Gpool in preference to G5.

When *puse* is a matrix, individual-specific allele frequencies are used (normally these would be population-specific, such as the breed or strain, including crosses). The calculations follow the method of Auvray *et al.* (2014) which is based on Harris and Johnson (2010).

When a PCA is requested, the eigenvalues displayed give the proportion of variation in G5 (the GRM) that is explained by each component (rather than the proportion of variation in the genotypes).

Some summary information is output. Gpool is used for the self-relatedness regression when samptype is "pooled".

Details: The function also produces a set of output files, as detailed below. If *withPlotly* is TRUE and both the plotly and heatmaply packages are available, interactive plotly plots are produced for some of the plots.

Co-call-<sfx>.png is a histogram of co-call rates (the proportion of SNPs with a call in both of a pair of samples) for all sample pairs. Bimodality may indicate that the samples belong to ≥2 genetically divergent clusters (with low co-call rates for pairs from different clusters). This plot is not produced if *calclevel*=1 or if there are more than √(2**.Machine$integer.max*) individuals.

MAF<sfx>.png is a histogram of the MAFs for the subset of SNPs used (if not all SNPs).

HighRelatedness<sfx>.csv contains pairs of samples, their G5 relatedness (G5rel) and self-relatednesses (SelfRel1 and SelfRel2), where the relatedness is > *hirel.thresh* (default value of *hirel.thresh* is 0.9).

Heatmap-G5<sfx>.png is a 'heatmap' plot using G5 relatedness (Gpool for *samptype* = "pooled"). Relatedness values are coloured from white to red (lowest to highest relatedness). Samples are ordered by a dendrogram representation of hierarchically clustering a distance matrix of the GRM. This plot is not produced if *npc*≤0. If *fcolo* for the relevant individuals has more than one colour, colour bars are added to the plot.

ColourKey<sfx>heatmap.png is a colour key for the heatmap.

HeatmapOrder<sfx>.csv contains a list of the samples in the order they are plotted on the heatmap. rowInd is the index values (written on the heatmap plot), seqIDInd is the position of the individual in seqID.csv; seqID is also included. For "standard" cases, where all seqID samples are included, the values of rowInd and seqIDind will be the same.

Heatmap-G5<sfx>.html is a plotly version of *Heatmap-G5<sfx>.png*. *plotly.group* and *plotly.group2* are not used.

Gcompare<sfx>.png is a plot comparing relatedness estimates for G1, G3 and G5. The lines of equality (y=x) are shown in red.

G<sfx>-diag.png is a plot of diagonal elements (self-relatedness estimates) of G4 against those of G5 (illustrating the effect of correcting for depth). If *samptype* is "pooled" then a comparison of diagonal elements of G4, G5 and Gpool is produced. The line of equality (y=x) is shown in red.

G<sfx>-diag.html is a plotly plot of diagonal elements of G4 against those of G5. Produced if *withPlotly* is TRUE. If *samptype* is "pooled" then Gpool is used instead of G5.

G<sfx>diagdepth.png is a plot of diagonal elements of G5 (or Gpool if *samptype* is "pooled") against the logged sample depth. We do not expect there to be a relationship between these variables (unless planned) so this serves as a diagnostic for e.g. non-Mendelian SNPs and/or the assumption of random sampling of alleles during sequencing.

G<sfx>diagdepth.html is a plotlly version of *G<sfx>diagdepth.png*.

PC1v2G5<sfx>.png (if |*npc*|>0) is a plot of 2nd versus the 1st principal components. Points are plotted with open (if 100 or more samples) or closed circles. If only one component was requested, a histogram of the 1st component is produced.

PC1v2G5<sfx>.html is the plotly version of *PC1v2G5<sfx>.png.*

PCG5<sfx>.pdf (if |*npc*|>2) is a scatterplot matrix of the first |*npc*| principal components.

PC1vInb<sfx>.png (if |*npc*|>0) is a plot of estimated inbreeding (using G5 or Gpool) against the 1st principal component. An unexpected trend may indicate allele sampling does not follow the analysis model.

PC1vDepth<sfx>.png (if |*npc*|>0) is a plot of sample depth against the 1st principal component, produced when gform is not "chip". An unexpected trend may indicate that the data contain artefacts.

MDS1v2G5<sfx>.png (if |*npc*|>0) is a plot of 2nd versus the 1st principal coordinates (multidimensional scaling axes).

MDS1v2G5<sfx>.html is the plotly version of *MDS1v2G5<sfx>.png.*

There is a vector *fcolo* (length *nind*) of colours to be used for the individuals in these plots. It defaults to all black, but can be set after sourcing the program (and/or running GBSsummary) and before calling *calcG*.

Calculate the genomic self-relatedness (*calcGdiag*)

A function, *calcGdiag*, for calculating the genomic self-relatedness using the KGD method, is defined. This will be faster than using calcG and extracting the G5 diagonal, but also has less options and diagnostics.

Usage: calcGdiag(snpsubset, puse, indsubset, depth.min=0, depth.max=Inf, quiet=FALSE)
Arguments:

snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of the SNPs to use in the calculation. The default is to use all SNPs.
puse	a vector or matrix of (reference) allele frequencies to use in the calculations. The default is to use allele frequencies calculated on the basis of allele counts. See the *puse* argument for *calcG* for further details.

indsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals for which self-relatedness will be calculated. The default is to calculate for all individuals.
depth.min	The minimum depth for a SNP result for an individual to be used.
depth.max	The maximum depth for a SNP result for an individual to be used.
quiet	Set to TRUE to reduce the reporting from the function. The default is FALSE.

Value: a vector of self-relatedness estimates (diagonal elements of G5).

Calculate the genomic inbreeding using merged *samples* (*Inbsampled*)

This function merges results based on the IDs provided and calculates inbreeding (diagonal of G minus 1) based on a single read per SNP and sample (using *samples* created within *GBSsummary* if *outlevel* > 7, or by *redosamples*). An example of where this might be used is when the same sample has multiple GBS results and the user wishes to compare the inbreeding for the sample with that calculated with only 1 read used per GBS result, in case reads within a result are not binomially sampled.

Usage: Inbsampled(mergeIDs, snpsubset, puse=p)
Arguments:

mergeIDs	a vector of identifiers for all *nind* samples, such that samples that have the same identifier are to be merged.
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of the SNPs to use in the calculation. The default is to use all SNPs.
puse	a vector of (reference) allele frequencies to use in the calculations. The default is to use allele frequencies contained in *p*.

Value: a vector of inbreeding estimates for all *nind* samples. Samples with the same value of *mergeIDs* will have the same inbreeding estimate.

Convert to Weir-Goudet GRM (*G5toDAWG*)

A function, for converting a G5 object that has been calculated using *calcG* with *puse* being 0.5 for all SNPs into a depth-adjusted version of the Weir-Goudet GRM (Weir & Goudet, 2017).

Usage: G5toDAWG(Guse)
Arguments:

Guse	a GRM calculated with *calcG* and with *puse* being 0.5 for all SNPs

Details: A matrix with the same dimensions as *Guse* is returned.

Create combinations of sample pairs for joining in plots (groupjoins)

Sets up a data frame to control lines joining points in a call to *GRMPCA*.
Usage: groupjoins(groupid, firstonly=FALSE, stdpal= "rainbow", hclpals=character(0), groupsort=FALSE)
Arguments:

groupid	a grouping label for the set of samples being considered. For use with *GRMPCA*, this is to be in the same order as the samples given to *GRMPCA*.
firstonly	if TRUE, joins will only be from the first sample found in a group to each other sample, otherwise joins will be between all pairs in a group
stdpal	colours to use if *hclpals* is empty. Can be either a vector of colours or one of the keywords “rainbow”, “heat”, “terrain”, “topo”, “cm” referring to the corresponding R colour palette.
hclpals	a character vector containing colour palettes to use from hcl.pals() (instead of rainbow colours). Ignored if hcl.colors is not available. If more than one palette is specified, they are used sequentially and evenly, but there is no check for duplicated colours across the palettes.
groupsort	specifies whether the *groupid* levels are sorted before assigning colours. The default is FALSE which uses the order encountered in groupsort

specifies whether the *colgroup* levels are sorted before assigning colours. The default is FALSE which uses the order encountered in *colgroup*.

Details: A data frame with columns *ind1* (index of first individual in *groupid*), *ind2* (index of 2nd individual) and *joincol* (colour to use for line drawing between individuals).

PCA plots from a GRM or PCA (GRMPCA)

A function to produce PCA plots from a GRM or directly from a set of principal components. This function allows more control over the plots than those produced within calcG. This function is still under development and its functionality may change in the future.

Usage: GRMPCA (Guse, PCobj=NULL, npc=2, npcxtra=0, pcasymbol=0, plotname="PC", plotsfx="", pcacolo, plotord=NULL, legendf = NULL, legendpos = "", move.factor=0.05, cex.plotsize=1, softI=FALSE, use.irlba=FALSE, joinset=NULL, ...)

To be documented.

Arguments:

Guse …	
joinset	indexes of points to be joined in the plot of PC2 vs PC1. A data frame with columns *ind1* (index of first individual), *ind2* (index of 2nd individual) and *joincol* (colour to use for line drawing between individuals). Usually created with *groupjoins*.

Output genomic relationship matrix (*writeG*)

A function, *writeG*, for saving genomic relationship matrices, is defined.

Usage: writeG (Guse, outname, outtype=0, indsubset, IDuse, metadf=NULL, gzip=FALSE, diag=FALSE)

Arguments:

Guse	the G matrix of relationships to output, should be a square matrix, or a list containing an element G5 (for *outtypes* 1 to 5) and/or PC (for *outtype* 6) and optionally *cocall* and/or *nsnps* (for *outtype* 7). The list format for Guse would normally be the output of *calcG*.
outname	text used in the naming of the output file(s)
outtype	constant or vector containing the type(s) of output required. If *outtype* contains any of the following values, the corresponding output is produced: 1 an R datasets file containing the G matrix and corresponding *seqID* 2 a .csv file (gzipped if gzip=TRUE and data.table is installed) containing the G matrix with row and column headings 3 a .csv file containing the G matrix in "long" format, i.e. one row for every (unique) relationship pair, but not including selfs (unless *diag* is set to TRUE); columns are IDs of first and second individual, followed by the relatedness value 4 a .csv file containing inbreeding for each individual; first column contains IDs, second column contains inbreeding estimates 5 two tab delimited files (.tsv) for input into the t-SNE interactive browser at http://projector.tensorflow.org/ (allows exploration of dimension-reduced data from the PCA or t-SNE methods). 6 a .csv file containing the principal components (requires *Guse* to be a list with element PC) 7 grm and id files suitable for use with GCTA software. The grm file is gzipped if *gzip* is TRUE and data.table is installed.
indsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals in the G matrix. The default assumes all individuals.
IDuse	a vector of IDs to use in the output, corresponding to the order in *Guse*, the default is to use values of *seqID* as the identifiers (in which case seqID must exist)

metadf	a data frame with the same number of rows as the G matrix, containing sample information to pass to the t-SNE browser (*outtype* 5) or the PC file (*outtype* 6).
gzip	if TRUE and data.table is installed, the G matrix csv files (*outtype* 2 and 3) will be gzipped (.gz file). The default is FALSE.
diag	if FALSE (the default), *outtype* 3 produces a file without selfs (the GRM diagonal). Setting *diag* to TRUE will include the self-relatedness values.

Details: One or more files are written to the default directory, according to *outtype*:

<outname>.RData an R data file containing the G matrix and corresponding *IDuse* values, produced when *outtype* contains a 1. The G matrix is named based on the object specified in *Guse*, removing text up to $ and from [, if either of these are present. As an example using *writeG(Gfull$G5[1:100,1:100],outtype=1)* will result in the G matrix being named G5.

<outname>.csv a csv file containing the G matrix, produced when *outtype* contains a 2. The first column is labelled with the name of the object passed to *IDuse* and contains the values of *IDuse*. The other columns are labelled with the values of *IDuse*.

<outname>-long.csv a csv file containing the unique relatedness values, one row for every pair of individuals (including selfs), produced when *outtype* contains a 3. The columns are labelled id1, id2 (lower case to avoid warning messages when opening with Excel) and rel. *IDuse* is used for the ID values.

<outname>-Inbreeding.csv a csv file containing inbreeding values (self-relatedness minus 1), produced when *outtype* contains a 4. The first column is labelled with the name of the object passed to *IDuse* and the second column as Inbreeding. *IDuse* is used for the ID values.

<outname>-pca_vectors.tsv a tsv file containing the G matrix in a format suitable for the t-SNE browser, produced when *outtype* contains a 5.

<outname>-pca_metadata.tsv a tsv file containing sample information (from *metadf,* or *IDuse* if *metadf* is NULL) in a format suitable for the t-SNE browser, produced when *outtype* contains a 5.

<outname>-PC.csv a csv file containing principal components, produced when *outtype* contains a 6 and *Guse* is a list containing PC (*Guse* is assumed to be the output from *calcG*). The first columns are from *metadf*, if given, or the object passed to *IDuse*. Subsequent columns are the principal components, labelled PC1, PC2 etc.

<outname>-gcta.grm.id a tab delimited file containing a column of 0's and a column of the IDs of the individuals in the corresponding .grm file. There are no column headers

<outname>-gcta.grm a tab delimited file with the first two columns containing index (position in the .id file) of each pair of individuals, a column of cocall counts and a column of relatedness values. The fille is gzipped (with .gz appended to the name) if *gzip*=TRUE and data.table is installed. The cocall counts are the cocall counts used to construct the GRM if *cocallout*=TRUE is used in *calcG* and *Guse* is the output object from *calcG*. When *cocall*=FALSE is used with *calcG*, the number of SNPs is used as the cocall. If *Guse* is a matrix, *nsnps* is used as the cocall. There are no column headers.

Relatedness comparison function (*GCompare*)

This is a function to help make comparisons between different estimates of relatedness on the same set (or overlapping subsets) of individuals. These different estimates may come from different genotyping technologies (e.g. SNP chip vs GBS), different protocols (e.g. GBS with different restriction enzymes, different levels of multiplexing samples, different SNP callers) or using different SNP filters.

The program inputs a set of (genomic) relationship matrices (GRMs) and a corresponding set of individual IDs. The output is a set of scatterplots (possibly as scatterplot matrices) and corresponding regression output. The relatedness estimates between each pair of (different) individuals for each pair of GRMs are compared, as are those for the self-relatedness estimates for each individual. For any pair of GRM, all individuals common to the GRM are used. If there are duplicated IDs within any set a warning is printed and only the first observation for the individual is used.

The information in the upper and lower panels is determined by the functions *regpanel* (for upper panel which normally shows regression results) and *plotpanel* (for lower panel which normally shows the scatterplot). These can be redefined before running GCompare (e.g. to change the colour of the line of equality, to draw other reference lines etc). Setting one of them to NULL will suppress that half of the scatterplot matrix. Setting *regpanel* to NULL will also supress the statistics being added to the output plots when there are only 2 G matrices.

Additionally, if the *MethComp* (Carstensen, 2015) R package is installed, there can be corresponding sets of plots using this package, with scatterplots of relatedness estimates below the diagonal, and 'Bland-Altman' (BA) plots (Altman and Bland, 1983) above the diagonal. The Bland-Altman plots have the differences on the vertical and the means on the horizontal access, for the two relatedness estimates. These plots take a lot more CPU time than the regression plots.

Usage: GCompare (Glist, IDlist, Gnames = paste0("G.",1:length(Glist)), plotname = "", whichplot="both", doBA=FALSE, ...)

Arguments:

Glist	a list of G matrices
IDlist	a list of ID variables, paired to the G matrices and in the same order as the data in the corresponding G matrix
Gnames	a set of labels to use for the G matrices (defaults to G1, G2, …)
plotname	text to use in the naming of output files
whichplot	variable to choose which plot types are produced, can be one of: "diag": compare diagonals (self-relatedness) "off": compare off-diagonals (relatedness between individuals) "both": compare both diagonals and off-diagonals. This is the default.
doBA	Additionally produce Bland-Altman plots. The default is FALSE.
…	Arguments to be passed to the plotting functions (e.g. col= for coloring). These need to be relevant to the plot types being produced (e.g. if a vector of colours, then it should not be used with *whichplot*="both").

Details: One or more plots are produced, depending on the options used. Regression statistics relating to each comparison are displayed. A set of ignorable warnings is issued.

Gcompare- <plotname>-diag.png a plot of the diagonal comparison(s). If more than 2 G matrices, this will be a scatterplot matrix with regression results in the upper matrix panels. A red line is drawn where values are equal.

Gcompare- <plotname>-offdiag.png a plot of the diagonal comparison(s). If more than 2 G matrices, this will be a scatterplot matrix with regression results in the upper matrix panels. A red line is drawn where values are equal.

GcompareBA- <plotname>-diag.png a scatterplot matrix BA plot of the diagonal comparison(s). The regression plots are in the lower diagonal and the BA plots in the upper diagonal. A grey line indicates equality (y=x for lower plots, y=0 for upper plots). The BA plots have 3 additional horizontal lines being the mean & mean ± 1.96sd ('95% limits of agreement).

GcompareBA- <plotname>-offdiag.png a scatterplot matrix BA plot of the off-diagonal comparison(s). See description of the BA plot for the diagonals for more details.

Bend a genomic relationship matrix (*Gbend*)

The function *Gbend* will 'bend' a square matrix to make it positive definite. The bending method used was proposed by Schaeffer (2013) and involves an eigen-decomposition of the input matrix, modification of the eigenvalues followed by a reconstruction. The method given here simply increases all eigenvalues below a threshold to that threshold. There are no guarantees to the performance of this method (use at your own risk!). The output matrix should be compatible with GBLUP methods.

Usage: Gbend (GRM, mineval=0.001, doplot=TRUE, sfx="", evalsum="free")

Arguments:

GRM	the square matrix to bend (usually a genomic relatedness matrix)
mineval	the lower threshold for eigenvalues. Values less than *mineval* are set at *mineval*. The default is 0.001.
doplot	If TRUE (the default), produce output plots (see below).
sfx	A suffix/label for the plot names and titles
evalsum	If "fixed", modified eigenvalues are rescaled to the original sum. The default is "free" (do not rescale).

Details: A matrix with the same dimensions as *GRM* is returned. If *doplot* is TRUE, three plots are produced:

Eigenvalues<sfx>.png a plot of the ordered original eigenvalues diagonal comparison(s). Negative eigenvalues are plotted in blue (positive values in black). A line is drawn at the *mineval* threshold.

Self-Bending<sfx>.png a plot of the diagonal values (self-relatedness values for a GRM) of *GRM* after vs before bending. The red lines shows where these are equal.

Rel-Bending<sfx>.png a plot of the off-diagonal values (between sample relatedness values for a GRM) of *GRM* after vs b before bending. The red lines shows where these are equal.

Output data in variant call format (*writeVCF*)

A function, *writeVCF*, for saving data in VCF format is defined.

Usage: writeVCF(indsubset, snpsubset, outname=NULL, ep=0.001, puse = p, IDuse, keepgt=TRUE, onlygt=FALSE, mindepth=0, allele.ref="C", allele.alt="G", GTmethod="observed", usePL = FALSE, contig.meta = FALSE, CHROM=NULL, POS=NULL, extrafields=list())

Arguments:

indsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals in to be output. The default assumes all individuals.
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of the SNPs to output. The default is to use all SNPs.
outname	base name of the output file which will have the extension ".vcf" appended. The default is "GBSdata".
ep	the probability of a sequencing error. Default to 0.001 (changed in v0.8.2).
puse	a vector of length *nsnps* of (reference) allele frequencies to use in the calculation of posterior genotype probabilities. The default is to use *p* (normally the allele frequencies calculated based on allele counts).
IDuse	a vector of IDs of length *nind* or the same length as *indsubset* to use in the output. The default is to use values of *seqID* as the identifiers (in which case seqID must exist)
keepgt	sets the genotype field (gt) to missing when *keepgt* is FALSE. The default is TRUE. This may be useful to force downstream analyses (e.g. Beagle 4.0) to use the likelihood field rather than the genotype.
onlygt	produces a vcf with only the genotype (gt) field when TRUE. The default is FALSE.

mindepth	the minimum depth for retaining values. Any results with a lower depth is set to missing for all fields except AD (allelic depth). The default value is 0 (don't set any values to missing).
allele.ref	reference allele symbol(s). Either a single character (used for all SNPs), a character vector of the same length as *snpsubset* (corresponding to each SNP in that list) or a character vector of length *nsnps* specifying the reference allele for all SNPs. If the input file was in TagDigger format, setting this to *refalleles* would be appropriate. The default is to use "C" for all SNPs being output.
allele.alt	alternate allele symbol(s) specified in the same manner as *allele.ref*. If the input file was in TagDigger format, setting this to *altalleles* would be appropriate. The default is to use "G" for all SNPs being output.
GTmethod	specifies how genotypes are derived. The default ("observed" is to use the raw (observed) genotype (regardless of depth). One other method, "GP", is available, which uses the genotype with the highest probability.
usePL	indicator which, if set to TRUE, will result in the output containing phred-scaled likelihoods instead of genotype likelihoods. The default is FALSE.
contig.meta	indicator which, if set to TRUE, will add contig meta info to the file (ID's only). Required for input into ANGSD. The default is FALSE.
CHROM	a vector of length *nsnps* containing chromosome labels for the SNPs. If not NULL, this will be used for the CHROM field, except when gform is "Tassel".
POS	a vector of length *nsnps* containing chromosome positions (in basepairs) for the SNPs. If not NULL, this will be used for the POS field, except when gform is "Tassel".
extrafields	a list of extra fields to include in the genotype result. Each field needs to have items *formatkey* (the information to be written in the FORMAT line), *fieldname* (the abbreviation for the field name, analogous to GT used for the genotype field), and *value* the value(s) for the field. This is expected to be a single value or a # samples x # SNPs matrix of values.

Details: A VCF format (https://samtools.github.io/hts-specs/VCFv4.3.pdf) file of the requested data is written. The file contains four or more fields of information relating to a genotype:

GT: the inferred genotype (0/0, 0/1, 1/1, and ./. for homozygous for reference allele, heterozygous, homozygous for alternate allele and missing, respectively)

GP: the three posterior genotype probabilities with priors calculated from the allele frequencies and assuming Hardy-Weinberg equilibrium, ordered corresponding to genotypes 0/0, 0/1, 1/1 (in GT format),

GL: (if usePL=FALSE) three $\log_{10}$-scaled likelihoods, calculated as in Li (2011), in the same order as GP,

PL: (IF usePL=TRUE) phred-scaled likelihoods (-10 * (GL – max(GL)) rounded to the nearest integer,

AD: allelic depth (read depth for reference and alternate alleles),

DP: read depth (both alleles)

as well as any fields defined in the extrafields parameter. If gform is "Tassel", then the CHROM and POS fields in the output are obtained from the input data. Otherwise, if CHROM and POS are given they are used for the CHROM and POS output fields, respectively. If netiher of these conditions holds, CHROM is specified as the SNP_Name and the position is numbered sequentially from 1. When CHROM and POS are available, the output is sorted by these values (CHROM and then POS). The variants are all denoted as C (REF allele) and G (ALT allele). Currently there is no facility for incorporating other genomic information passed either in the input file (e.g. if Tassel format) or as additional information.

If *gform* is "chip", some depths are infinite and *alleles* is not present (as is normally the case if the data are read in chip format) then the probability and likelihood calculations use maximum allelic depths of 500. It is also recommended that *ep* is set to 0, as this parameter related to allelic errors (sequencing) rather than genotype errors. Consider using *onlygt* = TRUE for chip data.

<outname>.vcf	a vcf formatted file of the data specified.

Output GBS data (*writeGBS*)

A function, *writeGBS*, for saving data is defined. Currently the only supported formats are the UNEAK, Tagdigger and chip formats (see the Input formats section).

Usage: writeGBS(indsubset, snpsubset, outname= "HapMap.hmc.txt", outformat=gform, seqIDuse=seqID, allele.ref="C", allele.alt="G")

Arguments:

indsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals to output. The default assumes all individuals.
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of the SNPs to output. The default is to use all SNPs.
outname	name of the output file. The default is “HapMap.hmc.txt”.
outformat	the format of the output file. The default value is *gform* (the format of the input file). Currently only the “uneak”, “TagDigger” or “chip” (not case-sensitive) formats will produce an output file. Any other value of *outformat* will produce a warning message.
seqIDuse	a vector of IDs of length *nind* or *indsubset* to use in the output (and which will be read as *seqID* if the file is read back in). The default is to use values of *seqID.*
allele.ref	the reference alleles. If a single value then this is used for all reference alleles. The default is “C”. Used when *outformat* is “TagDigger”.
allele.ref	the alternate alleles. If a single value then this is used for all alternate alleles. The default is “G”. Used when *outformat* is “TagDigger”.

Details: A data file with the specified format is written. For the uneak format, the function requires that the object *alleles* exists and that it corresponds to the genotype matrix (*genon*). It may be necessary to set *alleles.keep* to TRUE before data manipulation to ensure this is the case.

<outname>	a file of the data specified.

Gender prediction (*genderassign*)

The function *genderassign* can be used to predict gender using the methods described in Bilton *et al*. (2019). The assignment boundaries are specified by two functions *upperbounday*(x) and *lowerboundary*(x) where x represents the proportion of heterozygotes on the homogametic sex chromosome. These functions can be modified before using *genderassign*, but only *upperboundary* can be non-linear (not checked). Their initial values are:

```
upperboundary <- function(x){ 20*pmax(rep(0,length(x)),x)^2+0.2}
lowerboundary <- function(x){ 0.1 + x}
```

Usage: genderassign (ped.df, index_Y_SNPs, index_X_SNPs, sfx="", hetgamsex = "M", homgamsex = "F", hetchrom = "Y", homchrom = "X", dojitter = FALSE)

Arguments:

ped.df	a dataframe of individuals for gender prediction, as if read from a pedigree file (see Input formats section). This optionally contains variables Sex (with values M, F or U for male, female, unknown) and Relationship (character, e.g. "progeny", "sire" or "dam")
index_Y_SNPs	a vector of positions of SNPs on the homogametic sex chromosome (Y chromosome for X/Y systems) to use.
index_X_SNPs	a vector of positions of SNPs on the heterogametic sex chromosome (X chromosome for X/Y systems) to use.
sfx	text to be included in output file names
plotname	text to use in the naming of output files
hetgamsex	gender label for the heterogametic sex. The default is “M” (for males, assumes X/Y system). Use “F” for the Z/W system.
homgamsex	gender label for the homogametic sex. The default is “F” (for females, assumes X/Y system). Use “M” for the Z/W system.

hetchrom	chromosome label for the heterogametic sex chromosome. The default is "Y" (assumes X/Y system). Use "W" for the Z/W system.
homchrom	chromosome label for the homogametic sex chromosome. The default is "X" (assumes X/Y system). Use "Z" for the Z/W system.
dojitter	if set to TRUE, values on the y axis are jittered. The default is FALSE.

Details: Outputs a dataframe containing the input data frame (*ped.df*), the predicted gender, new_prop_X (the ratio of heterozygosity proportion of SNPs in *index_X_SNPs* compared to their expected proportions given the depth), proportion_SNPs_Y (proportion of SNPs in *index_Y_SNPs* with a result) and sampdepth (the mean sample depth for the individual). An output file and plot are also produced.

gender_prediction<sfx>.csv a .csv file containing the same information as the output dataframe.

GenderPlot<sfx>.png a plot of the results similar to Figure 1 of Bilton *et al.* (2019), where females are plotted in red, males in blue, unknowns in grey. The light blue shaded region indicates individuals predicted to be male, while the light red shaded region indicates individuals predicted to be female.

Breed prediction (*breedpredict*)

This function can be used to predict 'breed' (genetic group) based on the breed allele frequencies for each SNP. Currently the only available method is the 'regression' method of Kuehn (2011). The method uses raw genotypes as true genotypes (no depth adjustment) as depth adjustment is likely to have little impact on the prediction.

Usage: breedpredict (indsubset, snpsubset, breedaf, missaf="drop", method="reg")

Arguments:

indsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals to retain. The default assumes all individuals in *colobject*.
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of the SNPs to use in the calculation. The default is to use all SNPs.
breedaf	a matrix of dimensions *nsnps* by the number of breeds, containing breed-specific allele frequencies.
missaf	method to use for missing allele frequencies. The default ("drop") means that only those SNPs without all breed frequencies non-missing are used. An alternative option ("bmeans") is to use (unweighted) mean of the breeds with non-missing allele frequences for each SNP.
method	method to predict the breed. Currently only the "reg" method is available.

Details: Outputs a matrix (length(indsubset) by number of breeds in the breedaf matrix) containing predicted breed proportions.

Finplot functions (*finplot*, *HWsigplot*, *finclass*)

These functions allow different colours to be applied to the normal 'finplot'. A finplot refers to a plot of *HWdis* against *maf* for each SNP. The functions *finplot* (coloured by depth) and *HWsigplot* (coloured by the significance of a Hardy-Weinberg test) are used in the normal running of the GBS-Chip-Gmatrix.R code (perhaps via *GBSsummary*) – see Output - files, but they can also be used outside of this to give plots with other options specified. *finclass* allows colouring based on a factor or character variable.

Usage: finplot (HWdiseq=HWdis, MAF=maf, plotname="finplot", finpalette=palette.aquatic, finxlim=c(0,0.5), finylim=c(-0.25, 0.25))

Arguments:

HWdiseq	a vector of *nsnp* y values for the plot (normally a set of HW disequilibrium values). The default is to use *HWdis*
MAF	a vector of *nsnp* x values for the plot (normally a set of minor allele frequencies). The default is to use *maf*
plotname	The name of the .png output file. The default is "finplot"

finpalette	A set of 50 colours to use for portraying *snpdepth*. The default is to use *palette.aquatic* (grey to blue). Other inbuilt palettes are *palette.terrain* (from terrain.colors) and *palette.temperature* (blue through white to red).
finxlim	a numeric vector of length 2 with the x-coordinate limits. The default is (0,0,5) (the bounds for minor allele frequencies).
finylim	a numeric vector of length 2 with the y-coordinate limits. The default is (-0.25,0,25) (the bounds for Hardy Weinberg disequilibrium).

Details: A finplot is produced, coloured by *snpdepth* (mean depth per SNP), truncated at 256. *snpdepth* is mapped to *finpalette* in a non-linear way, such that there is more separation at lower depths. If *finpalette* contains colours close to white, the background of the plot is set to grey.

<plotname>.png	a plot of SNPs coloured by *snpdepth*. Normally a finplot.

Usage: HWsigplot (HWdiseq=HWdis, MAF=maf, ll=l10LRT, plotname="HWdisMAFsig", finpalette=palette.aquatic, finxlim=c(0,0.5), finylim=c(-0.25, 0.25), llname="-log10 LRT", sortord=ll)

Arguments:

HWdiseq	as in the *finplot* function
MAF	as in the *finplot* function
ll	a vector of *nsnps* significance statistics to use in colouring the plot. The default is to use $\log_{10}$ likelihood values from the likelihood ratio test (without adjustment for depth) of Hardy-Weinberg equilibrium.
plotname	The name of the .png output file. The default is "HWdisMAFsig"
finpalette	A set of 50 colours to use for portraying *ll*. The default is to use *palette.aquatic* (grey to blue). See the *finplot* function for other inbuilt palettes. The palette used by the normal running of GBS-Chip-Gmatrix.R is colorRampPalette(c("deepskyblue2","red"))(50).
finxlim	as in the *finplot* function
finylim	as in the *finplot* function
llname	a label to use to describe the colours used. The default is "-log10 LRT".
sortord	a variable of length *nsnps* for plotting order. The default is variable specified by *ll*. The effect of sorting is that higher values are plotted last, and therefore makes these less likely to be overplotted.

Details: A finplot, but coloured by *ll*, is produced. The main code uses this function with *ll=l10pstar*, *llname*="-log10p X2*" and *finpalette*=colorRampPalette(c("deepskyblue2","red"))(50)) (light blue to red).

<plotname>.png	a plot of SNPs coloured by *ll*.

Usage: finclass (HWdiseq=HWdis, MAF=maf, colobj, classname=NULL, plotname="finclass", finxlim=c(0,0.5), finylim=c(-0.25, 0.25))

Arguments:

HWdiseq	as in the *finplot* function
MAF	as in the *finplot* function
colobj	a list as produced by *colourby* (see: Colouring functions (colourby, changecol, fadeco, coloursub, colkey, collegend) specifying the colour for each SNP
classname	a label to use to describe the colours used. The default is to use the colour class name (*col.name*) from *colobj* if it has one, otherwise a label is not used.
plotname	The name of the .png output file. The default is "finclass"
finxlim	as in the *finplot* function
finylim	as in the *finplot* function

Details: A finplot, but with SNPs coloured by the *sampcol* item of *colobj*, is produced. This can be a useful graphic for displaying filtered and unfiltered SNPs, for example. The key is labelled by the *col.name* value from *colobj* if it exists.

<plotname>.png	a plot of SNPs coloured by *colobj$sampcol*.

HDplot function (*HDplot*)

This function produces an HDplot, as described by McKinney *et al*. (2017) with the proportion of (observed) heterozygous individuals on the x axis, and the read-ratio deviation (from 1:1) is plotted on the y axis. The function allows different colouring options (see details below). The function saves some intermediate variables to the workspace to save re-calculation when requesting another colouring scheme (remove the variable *HD.saved.KGD* if the function is to be applied to different data).

Usage: HDplot (plotname="HDplot", colourtype = "depth", finpalette=palette.aquatic, HDxlim=c(0,1), HDylim=c(-Inf, Inf), HDcol=NULL, sortcol = "asc")

Arguments:

plotname	The name of the .png output file. The default is “HDplot”
colourtype	specifies the colouring scheme, when HDcol is NULL. Can be one of “depth” (colour by mean read depth as in the *finplot* function), “HW” (colour by the raw Hardy-Weinberg significance) or “HW*” (colour by the depth-adjusted Hardy-Weinberg significance). The latter two schemes colour in the same was as in the *HWsigplot* function. A legend is also added to the plot.
finpalette	A set of 50 colours to use for portraying the information specified by *colourtype*. The default is to use *palette.aquatic* (grey to blue). See the *finplot* function for other inbuilt palettes. Use colorRampPalette(c("deepskyblue2","red"))(50) when *colourtype* is “HW” or “HW*” to match the HWsigplot(s) from the normal running of GBS-Chip-Gmatrix.R.
HDxlim	a numeric vector of length 2 with the x-coordinate limits. The default is (0,1) (the bounds for heterozygosity).
HDylim	a numeric vector of length 2 with the y-coordinate limits. The default is (-Inf,Inf) in which case the data bounds are used. If the limits are set inside the data bounds, points outside the limits are plotted as “^” and “v” on the closest limit.
HDcol	a character vector of length *nsnps* specifying the colour for each SNP. If given, these colours override the action of *colourtype*. The default is NULL.
sortcol	an character option to plot the points in sorted colouring variable order (when *HDcol* is NULL), either ascending (“asc”) or descending (“desc”). Any other value will plot in data order. Points plotted last will be more noticeable.

Details: An “HDplot”, with SNPs coloured as specified is produced. This can be a useful graphic for detecting the presence of paralogous loci and for determining whether a SNP filter to be used will likely remove those SNPs.

<plotname>.png	an HDplot of SNPs.

Miscellaneous functions

These are functions to help make manipulations of the data easier.

Extract off-diagonal values from a square matrix (upper.vec)

This function extracts the upper triangular values from a square matrix and places them in a vector. This function is used by other functions (usually for comparing matrix values).

Usage: upper.vec (sqMatrix, diag=FALSE)

Argument:

sqMatrix	a square matrix
diag	if FALSE only include off-diagonal values, if TRUE include diagonal values as well

Details: Outputs numeric vector of upper triangular values, ordered by row then column.

Show the first rows and first columns of an object (corner)

This function prints the first rows and columns of an object with 2 dimensions. It is similar to the *head* function, but that only subsets rows (for 2-dimensional objects).

Usage: corner (mtx, size=6L)

Argument:

mtx	a 2-dimensional object (usually a matrix or data frame)
size	the number of rows and columns to display. The default is to display 6 rows and columns.

Details: displays the first rows and columns of the object.

Add the y=x (equality) line to a plot (eqline)

This function adds a y=x line to a (user-specified) plot. The function is the same as R's *abline*, but with different defaults: a=0,b=1,col="red",lwd=2.

Row and column summary functions

These functions extend the range of row and column summary functions available from base R. Unlike the base R functions, they do not provide any extensive checking. The extended summary functions are max, min, which.max, which.min. The new functions are named colMax, rowMax, colMin, rowMin, colwhich.Max, rowwhich.Max, colwhich.Min, rowwhich.Min.

Usage: <function> (x, na.rm=FALSE)

Arguments:

x	a 2-dimensional array
na.rm	Should NA's be removed before performing the summary. The default is FALSE.

Details: gives a vector of the summary data for each row (for 'row' functions) or column (for 'col' functions).

Extracting first 'field' from seqID (seq2samp, seq2samp1)

This function extracts the text from a character variable. *seq2samp1* is an earlier version of *seq2samp* that only returns the text that precedes a specified delimiter but should no longer be needed (this can be achieved with *seq2samp*) and is likely to be discontinued in the future. The normal use of this is to extract the first 'field' from *seqID*, when *seqID* contains information about the sequencing process separated by a delimiter (e.g. an underscore), and the first piece of information is a sample identifier. The sample identifier should not contain the delimiter.

Usage: seq2samp (seqIDvec=seqID, splitby="_", nparts=NULL, dfout=FALSE, ...)

Arguments:

seqIDvec	a character vector, usually of *seqID*s. The default is *seqID*
splitby	the character to delimit the end of the text to be extracted. The default is an underscore ("_")
nparts	Either NULL or (coerced to) an integer ≥2. This is the number of parts to split the *seqIDvec* into. If there are more *splitby* characters than *nparts* - 1, then the extra parts are combined as the first element, i.e. *splitby* can be within the first part. The default (NULL) returns the text before the first occurrence of the *splitby* character.
dfout	determines whether the output is a data.frame of all parts, named V1, V2 etc, (*dfout*=TRUE) or of just the first part (*dfout*=FALSE, the default)
…	further arguments to be passed to strsplit (e.g., using fixed=TRUE will treat *splitby* as plain text rather than a regular expression), used only if *nparts* is NULL.

Details: Outputs a character vector or data.frame of *nparts* character variables.

Usage: seq2samp1 (seqIDvec=seqID, splitby="_", …)

Arguments:

seqIDvec	a character vector, usually of *seqID*s. The default is *seqID*
splitby	the character to delimit the end of the text to be extracted. The default is an underscore ("_")
…	further arguments to be passed to strsplit (e.g., using fixed=TRUE will treat *splitby* as plain text rather than a regular expression)

Details: Outputs a character vector.

Colouring functions (colourby, changecol, fadeco, coloursub, colkey, collegend)

Some functions are provided to help specify plotting colours based on a factor or variable. *colourby* creates a set of colours (and possibly symbols), *changecol* allows these to be modified, *fadeco* makes the colours transparent, *coloursub* reduces the set to the specified individuals and *colkey* plots a key to the colours and symbols (if relevant). *collegend* is a function to facilitate the placement of legends on plots using the information in an object created by *colourby*. There are also functions *fade* and *darken* for modifying a vector of colours (not documented).

Usage: colourby (colgroup, nbreaks=0, col.name=NULL, symbgroup=NULL, symb.name=NULL, groupsort=FALSE, grouporder=NULL, maxlight=1, alpha=1, reverse=FALSE, symbset=NULL, stdpal= "rainbow", hclpals=character(0), pal.upper=1, nacolour="black")

Arguments:

colgroup	a vector or factor, whereby each level (unique value) will be given a different colour (when *nbreaks*=0) or used to create colour groups (when *nbreaks* >0).
nbreaks	integer. When nbreaks > 0, *colgroup* is interpreted as a continuous variable and values are placed into approximately *nbreaks*-1 groups. The default for *nbreaks* is zero (*colgroup* is treated as a factor with distinct levels).
col.name	a descriptor for the colouring variable.
symbgroup	a character vector or factor, whereby each level (unique value) will be given a different symbol (if symbset is not NULL). If there are insufficient symbols in *symbset* it will be augmented with unspecified symbols from 1,2, … .
symb.name	a descriptor for the symbol variable.
groupsort	specifies whether the *colgroup* levels are sorted before assigning colours. The default is FALSE which uses the order encountered in *colgroup*.
grouporder	a vector of integers specifying an ordering of the *colgroup* levels before assigning colours. This is ignored if its length is not the number of *colgroup* levels. Normally used with *groupsort* set to TRUE. The default is NULL, no reordering.
maxlight	A value in (0,1] limiting the 'lightness' of the colours assigned
alpha	the degree of transparency, where 1 (the default) is the full colour and 0 is fully transparent (no colouring).
reverse	if TRUE, the colour order will be reversed. The default is FALSE.
symbset	A set of numeric symbol codes (as used with the pch parameter in R plotting) to assign to the groups. They are used for the grouping given in *symbgroup*, if specified, otherwise for the grouping specified in *colgroup*. These are recycled if needed. The default is NULL in which case no symbols are assigned.
stdpal	colours to use if *hclpals* is empty. Can be either a vector of colours or one of the keywords "rainbow", "heat", "terrain", "topo", "cm" referring to the corresponding R colour palette.
hclpals	a character vector containing colour palettes to use from hcl.pals() (instead of rainbow colours). Ignored if hcl.colors is not available. If more than one palette is specified, they are used sequentially and evenly, but there is no check for duplicated colours across the palettes.
pal.upper	upper boundary for the fraction of hclpals colour range that is used. The default is 1 (use the full range). This can be used to prevent using lighter colours when these are at the top end of the palette range.

nacolour	the colour to be used for when *colgroup* is NA. The default is "black".

<u>Details</u>: Outputs list with three elements:

collabels	The discrete values in *colgroup* or (numeric) midpoints of the group boundaries when *nbreaks*>0.
collist	The set of colours corresponding to each element of *collabels*
sampcol	The set of colours in *collist*, corresponding to *collabels*.
col.name	The value of *col.name*. Only present if specified in the call.
symblabels	The discrete values in *symbgroup*. Only present if *symbgroup* is specified in the call.
symblist	The set of symbols (numeric) corresponding to each element of *collabels* (included only if *symbset* is not NULL)
sampsymb	The set of symbols in *symblist*, corresponding to *colgroup* (included only if *symbset* is not NULL).
symb.name	The value of *symb.name*. Only present if specified in the call.

When *hclpals* is empty, the function chooses a set of colours from the palette specified by *stdpal*. The default (*stdpal*="rainbow") is colours that span the rgb range. When there are more than 8 levels and if *hclpals* is empty, every 2nd colour is made greyer, to help distinguish the colours. Normally the function is applied to samples, with *colgroup* of length *nind* corresponding to each *seqID*, although it can be applied to other items, e.g. to the SNPs.

<u>Usage</u>: changecol (colobject,colposition,newcolour)
<u>Arguments</u>:

colobject	a list that was created with *colourby*
colposition	the (set of) position(s) (integer(s)) in the *collabels* (and *collist*) element of *colobject* to be changed
newcolour	the (set of) new colours (character (vector)) to use in the *colposition* position(s) of the *collist* element of *colobject*

<u>Details</u>: Outputs a list with the same structure as *colobject*. The number of elements in *colposition* and *newcolour* should match.

<u>Usage</u>: fadeco (colobject, alpha)
<u>Arguments</u>:

colobject	a list that was created with *colourby*
alpha	the level of opacity, where 0 is no colour and 255 is full colour. If *alpha* ≤1 it is taken as the opacity fraction (opacity is 255 x *alpha*).

<u>Details</u>: Outputs a list with the same structure as *colobject*.

<u>Usage</u>: coloursub (colobject,indsubset)
<u>Arguments</u>:

colobject	a list that was created with *colourby*
indsubset	a vector of integers (between 1 and *nind,* inclusive) of the individuals to retain. The default assumes all individuals in *colobject*.

<u>Details</u>: Outputs a list with the same structure as *colobject* with the set of individuals subset to those in *indsubset*. The elements *sampcol* and *sampsymb* are subset to *indsubset* while the levels and labels of the colours and symbols (if present) are subset to those used.

<u>Usage</u>: colkey (colobj, sfx="", srt.lab=0, plotch=16, horiz=TRUE, freq=FALSE)
<u>Arguments</u>:

colobj	a list that was created with *colourby* (or *changecol*)
sfx	text to be used in naming the output file
srt.lab	string rotation setting for the labels. Common values are 0 (the default, horizontal text) or 90 (vertical text).
plotch	an integer or vector of integers of length equal to the length of *colgroup* in *colobj*. Denotes the plotting symbol(s) to use. Defaults to 16 (solid circle). If *colobj* does not include *symblabels* but includes a *symblist*, then that is used instead.

horiz	a logical value. If TRUE (default), the symbols are drawn horizontally with the text below. If FALSE, the symbols are drawn vertically with the text to the right.
freq	if TRUE, provides a table(s) of frequency counts for each level of the colouring and (if present) symbol variables. The default is FALSE.

Details: A plot is produced. The pointsize used for the plot is *cex.pointsize* * default pointsize.

ColourKey<sfx>.png A plot showing a set of points or a colour raster, coloured with *colobj$collist* colours and labelled with *colobj$collabels*. A raster is displayed if there are at more than two colours and *colobj$collabels* is numeric. The points are set by *plotch* or the symbols in *colobj$symblist* if it is the same length as *colobj$collist* and there is no *colobj$symblabels*. If *colobj* includes *symblabels* then an additional key is plotted showing the symbols and corresponding labels (in black). For both colours and symbols, if there is a corresponding name in *colobj* (*col.name* and *symb.name*, respectively) then they are used as titles.

Usage: collegend (colobj, legpos="topleft", plotx=NULL, ploty=NULL, cex.leg=0.9, legendsym = 1)

colobj	a list containing the legend information, normally created with *colourby* (or *changecol*)
legpos	the position of the legend, in a form acceptable as the *x* argument of the base R *legend* function.
plotx	(optionally) the x (horizontal axis) values for the plot. When both plotx and ploty are given, the legend is checked to see if it overlaps any if so, a warning is issued.
cex.leg	the relative size of legend text. The default is 0.9.
legendsym	the symbol to use in the legend unless specified in *colobj*.

Details: This function is still under development and its functionality may change in the future. A legend is added to the current plot for the colours and symbols (if relevant) used. If *colobj$collabels* is numeric, the colours are interpreted as a colour gradient and a raster is plotted as the colour legend. If *colobj* also contains symbol labels (symblabels) which are not a 1-1 match with the colours then an additional legend is plotted for the symbols. This is added beside the first legend with some attempt to position it where there is minimal data. The *collegend* statement is to be used where an R legend statement could be given.

Label positioning utility function (labelpos)

Used by GRMPCA.

Usage: labelpos (x, y, xrange=range(x), yrange=range(y), gapp=0.02, neighbourp=0.1, xpd=FALSE)

Pedigree analysis (GBSPedAssign.R)

The R *source* command is used to invoke the GBSPedAssign.R code. This code will (optionally, see *functions.only*) run the parentage analysis to verify parents (if given) or assign parents (if parent groups are given) after defining a set of functions, The following table shows variables that are required to or can optionally be set are shown in Table 3.

Table 3 Variables that can be set in the calling program for undertaking a parentage analysis.

Variable	Description
pedfile	Name of file containing pedigree and/or parent group information
groupsfile	Name of file containing which individuals are in which parent groups. This file is ignored if *matesfile* is defined.
matesfile	Name of file containing which mate pairs are in which parent groups
GCheck	The name (as a string) of the G matrix to use for parent verification or assignment This must be set before calling GBSPedAssign.R.
indsubset	The subset of individuals used to calculate the matrix specified in *GCheck*.

rel.thresh	The relatedness threshold to use for parent verification or assignment, if the corresponding parent sex-specific threshold (*rel.threshF* or *rel.threshM*) has not been set. This has a default value of 0.4.
rel.threshF	The relatedness threshold to use for father verification or assignment. This has a default value of *rel.thresh*.
rel.threshF	The relatedness threshold to use for mother verification or assignment. This has a default value of *rel.thresh*.
mindepth.mm	Minimum depth to be used for calculating mismatch proportions in parent matching. Default is 1 (use all results).
snpsubset	The subset of SNPs to be used for calculating mismatch rates or for bootstrapping (usually the same set as used for calculating calculate the matrix specified in *GCheck)*. Default is all SNPs.
emm.thresh	The excess mismatch rate threshold to use for parent assignment. This has a default value of 0.01.
emm.thresh2	The excess mismatch rate threshold to use for parent-pair assignment. This has a default value 2 x emm.thresh.
doublemm	When true, count cases where both sire and dam are mismatches as 2 in trio mismatch counts (and expected mismatch counts). TRUE is the preferred setting, although the default is FALSE for consistency with earlier versions.
emmdiff.thresh2	The excess mismatch rate difference (from that for the most related father and most related mother) threshold to use for suggesting an alternate parent-pair assignment. This has a default value of 0.
inb.thresh	The lower threshold for the difference between parent relatedness and twice the estimated inbreeding to exclude a parent-pair match with the inbreeding check. This has a default value of 0.2.
minr4inb	The lower threshold on parent relatedness to exclude a parent-pair match with the inbreeding check. This has a default value of NULL (no minimum).
boot.thresh	If the relatedness with the 2nd best parent is within *boot.thresh* of that for the best parent, a bootstrapping procedure will be invoked to further compare these possible matches. Default value of 0.05.
depth.min	Minimum mean depth of SNPs to be used for bootstrapping. Default value is 0.
depth.max	Maximum mean depth of SNPs to be used for bootstrapping. Default value is 0.
puse	Allele frequencies to be used when bootstrapping. Default is to use *p*.
nboot	Number of bootstrap replicates. Default value is 1000.
boota.thresh	The upper threshold on bootstrap reliability for excluding a parent match with the bootstrapping check. This has a default value of 99.
allow.selfing	If TRUE, the same individual can be assigned as male and female parent (default is FALSE).
matchmethod	The method used to find the best 2 matching parents (fathers and/or mothers). The default value is "rel" where the maximum relatedness is used. The alternative is "EMM" where minimum EMM is used. At this stage bootstrapping and alternate assignments are based on using "rel" so may not give sensible results with "EMM".
allow.selfing	If FALSE (the default), then results (when parent matching in groups) are adjusted to prevent the same individual being the best matching mother and father.

This program uses a relatedness matrix and excess mismatch rate (EMM) results for pedigree (i.e. parent) analysis using the methods described in Dodds *et al.* (2019). There are three different types of analyses that are possible: 1) verify given pedigrees, 2) find the best matching parents from lists of potential fathers and of possible mothers (groups matching) and/or 3)) find the best matching parents from a list of mate pairs (mates matching). These tasks require a pedigree file (with name given in *pedfile*).

Father (Mother) verification is undertaken if the pedigree file contains a FatherID (MotherID) variable. If both FatherID and MotherID are in the pedigree file there is an additional test that the trio is consistent (using the trio excess mismatch rate). An individual may have missing values for one or both (if present) of FatherID and MotherID in which case the corresponding match result is missing (NA).

For parent matching a groups file (with name given in *groupsfile*) or a mate pairs file (with name given in *matesfile*) is also required. If there is a *matesfile*, these combinations are used for checking and *groupsfile* (if given) is ignored. See below for the formats for these files. Father (Mother) matching is undertaken if a groups file is given and the pedigree file contains a FatherGroup (MotherGroup) variable. These variables contain labels which refer to the possible set of parents, which are given against that label in the *groupsfile*. If both FatherGroup and MotherGroup are present, there is further testing for trio consistency. The Group fields are read as text fields. An individual may have missing values for one or both (if present) of FatherGroup and MotherGroup in which case the corresponding match result is missing (NA). If the individual's FatherGroup (or MotherGroup) has no records in *groupsfile*, the Father (or Mother) is not assigned. If there is a single possible parent for that group in *groupsfile*, that parent will be the "best" parent and there will be no 2nd best parent. Parent-pair matching is undertaken if a mates file is given and the pedigree file contains a MatesGroup field. This variable contains labels which refer to the possible parent-pairs, which are given against that label in the *matesfile*.

For parent matching, mismatch statistics are calculated for reporting and using, in addition to relatedness values, for assigning parentage. The 'raw' mismatch rate is the proportion of apparent (i.e. using observed genotypes) mismatches (i.e., genotypes inconsistent with parentage). 'Excess' rates are the differences between raw rates and rates that are expected given the genotype uncertainty due to the GBS process (manuscript in prep). A number of variables (see below) control how the mismatch rates are calculated and used. Mismatch rates are calculated for offspring-parent pairs and for offspring-parent trios (if matching to both parents). If both parents are being matched, the apparent parent-pair mismatch rates (offspring and parent genotypes incompatible) are given for each combination of the best two matching parents.

The following rules are implemented when individuals can be a father or a mother, but not both (*allow.selfing* is false). If an individual is the best matching male and female parent, then one of the 2nd best parents (male or female) will be promoted (the one that gives the better combined parent matching statistics, either relatedness or EMM depending on *matchmethod*). In addition, a promoted individual will be removed where it is a 2nd best match for the other parent type (and relevant bootstrap results are ignored). Single parent matching graphs are not redone.

Before calling the function *GBSPed* (or sourcing the R code with *functions.only* set to FALSE), the variable *GCheck* must be set to the name (as a string) of the G matrix to use. If this is for a subset of individuals, *indsubset* must be set to the indices of those individuals (as used in *calcG*). In addition, *rel.thresh* (and/or *rel.threshF* and/or *rel.threshM* for fathers and mothers, respectively) may be set to override the default relatedness value of 0.4 for declaring a parentage verification (or to allow parent assignment). A number of other variables control calculated results and reporting for parent matching. *mindepth.mm* may be set to override the default minimum depth (1) for a SNP for the individuals being compared when calculating (excess) mismatch rates for parentage matching. The default value is recommended for calculating excess rates, but raw rates are likely to be more useful when using a higher threshold. *snpsubset* may be set to indices of SNPs to be considered for use in calculating mismatch rates and for bootstrapping (see below, this will usually be the same subset as used for calculating the G matrix being used). The excess mismatch rate thresholds for declaring parentage are set by *emm.thresh* (parent-offspring pair; default value of 0.01) and *emm.thresh2* (parent-offspring trio; default value of twice *emm.thresh*). An alternative parentage is suggested when a possible pair (mother and father) have an excess mismatch rate that is lower than that for the best (i.e., most highly related) father and best mother by more than *emmdiff.thresh2* (default value of 0).

For parent pair matching (groups or mates matching), the estimated relatedness between the parent pairs (all four combinations of best and 2nd best matching fathers and mothers, or the best and 2nd best pairs, if *matesfile* is used) are calculated. The relatedness for the best matching pair of parents is compared with the estimated inbreeding for the individual. High values of parent relatedness (compared with the inbreeding of the individual) may indicate that one of the parents has been incorrectly assigned to a relative of the other parent. A parent-pair match will be excluded as a match if the parent relatedness exceeds offspring inbreeding by at least *inb.thresh* (default value 0.2).

A bootstrapping procedure is available to provide a metric on the closeness of parent-offspring match compared to that with the 2nd best parent. The procedure resamples SNPs (with replacement), recalculates the relatedness values (for the offspring and each of the two best parents) and reports the percentage of times that the best parent is still the better of the two among the bootstrap replicates. This should not be taken as a significance level test, as the resampled SNPS are not independent. As bootstrapping is quite time-consuming, it is invoked only when there are 2 possible parents with similar (within *boot.thresh*) parent-offspring relatedness values, and if the best parent exceeds the relatedness and excess mismatch thresholds. The number of bootstrap replicates is set by *nboot* (default value 1000). Three other variables (*depth.min, depth.max, puse)* mirror those used in calcG to allow the bootstrapping to calculate relatedness in the same way as was used for the G matrix being used in parentage assignment. These variables should be set to the same values as those used for calculating the G matrix. An assignment is flagged (see below) if the best parent is the better one in the bootstrap samples in less than *boota.thresh* percent (default value 99) of the replicates. Bootstrapping is not conducted if *matesfile* is used.

For groups matching, if some individuals are common to the father groups and mother groups (i.e. individuals may be a mother or a father) then it is possible that the best matching mother and best matching father are the same individual. If selfing is not possible (*allowing.selfing* is FALSE), then these results will be adjusted by choosing the better of the 2nd best matching parents as the parent for that gender.In this case the statistics for the 2nd best matching parent and bootstrap results are not given (NA). In addition, the *mmnum* variable for the replaced parent is also given as an NA (which gives a way to distinguish cases where this has happened from those where only one parent was possible). Plots from each single parent analysis are not updated with the selfing exclusions.

The output files contain variables to indicate whether the parentage should be accepted. These variables are called *FatherAssign* and *MotherAssign* for single parent matching of fathers and mothers, respectively. The codes used as values for these variables are shown in Table 4.

Table 4 Assign codes for FatherAssign and MotherAssign.

Assign code	Description
N	Relatedness estimate for best matching parent is below *rel.threshF* or *rel.threshM* (for fathers and mothers, respectively).
E	Excess mismatch rate for best matching parent exceeds *emm.thresh.*
A	Alternate assignment: the 2nd best parent appears acceptable. This parent has relatedness exceeding *rel.threshF* or *rel.threshM* (for fathers and mothers, respectively) and excess mismatch rate that is lower than *emm.thresh* when the best parent had excess mismatch rate exceeding this threshold.
B	Best matching parent is the better one in less than *boota.thresh* % of the bootstrap replicates.
Y	Best matching parent passes all assignment criteria

The variable for indicating whether a parent-pair match should be accepted is *BothAssign* and takes values as shown in Table 5.

Table 5 Assign codes for BothAssign.

Assign code	Description
N	Relatedness estimate for best matching parents are below *rel.threshF* and *rel.threshM* (for fathers and mothers, respectively).
M	Mother assigned, father not assigned.
F	Father assigned, mother not assigned.
E	Excess mismatch rate for best matching parent-pair exceeds *emm.thresh2*, except when one parent assigned and the other has an E code, then the parent assignment is made.
A	An alternate parent-pair appears acceptable. This pair has excess mismatch rate less than *emm.thresh2* and lower than that for the best parent-pair by more than *emmdiff.thresh2*. If the alternate pair also passes the other checks, the pair is indicated by the value of *Alternate*, e.g. a value of F1M2 indicates that the alternate pair is the best father and 2nd best mother.
B	At least one of the parents has a B code. (It may still be possible to assign the other parent).
I	The best parent-pair relatedness exceeds twice the offspring inbreeding by at least *inb.thresh*, and is above *minr4inb* (if that threshold has been set). An alternate pair may be indicated by the value of *Alternate*, similarly to the A code offspring.
Y	Best matching parent passes all assignment criteria

Where more than one of the assign codes is possible, the one that ranks the highest (in the order given in the above tables) is used.

This program outputs summary statistics and a number of files. The %s of verified fathers and mothers are given, as well as the mean relatedness estimates for matching and non-matching fathers and mothers. The files, where relevant, are as follows:

PedVerify.csv returns the pedigree file with additional columns, as shown in Table 6.The "Father" ("Mother") columns are present only if *FatherID* (*MotherID*) was in *pedfile*, while the "FandM" columns are present only if both *FatherID* and *MotherID* are in *pedfile*.

Table 6 Columns in PedVerify.csv in addition to those in the pedigree file.

Variable name	Description
Inb	The estimated inbreeding of the offspring
FatherRel	Relatedness estimate between individual and its specified father
FatherEMM	The specified father-offspring EMM
FatherMatch	TRUE if *FatherRel* > *rel.threshF* and *FatherEMM* < *emm.thresh*
FatherInb	The estimated inbreeding of *FatherID*.
MotherRel	Relatedness estimate between individual and its specified mother
MotherEMM	The specified mother-offspring EMM
MotherMatch	TRUE if *MotherRel* > *rel.threshM* and *MotherEMM* < *emm.thresh*
MotherInb	The estimated inbreeding of *MotherID*.
FandMEMM	The specified parent pair – offspring EMM
FandMmatch	TRUE if *FatherMatch* and *MotherMatch* are both TRUE and *FandMEMM* < *emm.thresh2*

FatherVerify.png is a scatterplot matrix showing *FatherRel, FatherEMM* (see above), the position of the individual in the pedigree file and the position of the recorded father in the pedigree file. The thresholds used are shown as dotted lines in the plots of the first two variables. The sample order variables can be helpful for detecting sample tracking issues (if the order in the pedigree file relates to the order samples are processed at a particular stage).

MotherVerify.png is a scatterplot matrix like FatherVerify.png but for mother verification.

RecFatherMatchesE.png is a plot (when *FatherID* is present) of the excess mismatch rate for *FatherID* against the estimated relatedness (*Fatherrel*). Points are coloured using *fcolo* and grey shading indicates the excluded fathers based on the thresholds used.

RecMotherMatchesE.png is the same as RecFatherMatches.png but for mother verification.

ExpMM-RecFather.png is a plot of the raw mismatch rate against the expected mismatch rate for *FatherID*. A red line shows where these are equal, and grey shading indicates the excluded fathers. Points are coloured using *fcolo* and the symbols indicate *FatherMatch*.

ExpMM-RecMother.png is the same as a ExpMM-RecFather.png but for mother verification.

ExpMM-RecBoth.png is a plot of the raw mismatch rate against the expected parent-pair mismatch rate. A red line shows where these are equal, and grey shading indicates the excluded parent-pairs. Points are coloured using *fcolo* and the symbols indicate *FandMMatch*.

FatherMatches.csv shows the results of the father matching. It returns the first two columns of the pedigree file with additional columns, as shown in Table 7.

Table 7 Columns in FatherMatches.csv in addition to IndivID and seqID from the pedigree file.

Variable name	Description
BestFatherMatch	IndivID of the father from the *FatherGroup* having the highest estimated relatedness to the individual (or lowest EMM, if *matchmethod* is “EMM”).
FatherMatch2nd	IndivID of the father from the *FatherGroup* having the 2nd highest estimated relatedness to the individual (or 2nd lowest EMM, if *matchmethod* is “EMM”)
Fatherrel	The estimated relatedness for *BestFatherMatch*
Fatherrel2nd	The estimated relatedness for *FatherMatch2nd*
Father12rel	The estimated relatedness between *BestFatherMatch* and *FatherMatch2nd.*
mmrateFather	The (raw) mismatch rate for *BestFatherMatch*
mmnumFather	The number of snps used to calculate *mmrateFather*
exp.mmrateFather	The expected mismatch rate for *BestFatherMatch*
mmrateFather2	The (raw) mismatch rate for *FatherMatch2nd*
exp.mmrateFather2	The expected mismatch rate for *FatherMatch2nd*
Fathersd	The bootstrap sd of *Fatherrel* values (for bootstrapped cases, the variable is present only if there are bootstrapped caess)
FatherReliability	The % of bootstrap results where *Fatherrel* > *Fatherrel2nds* (for bootstrapped cases, the variable is present only if there are bootstrapped caess)
FatherAssign	The code for father assignment.
FatherInb	The estimated inbreeding of *BestFatherMatch*

MotherMatches.csv shows the results of the mother matching (with columns as for FatherMatches.csv but for mothers instead of fathers).

BothMatches.csv shows the results of both father and mother matching (for individuals with both *FatherGroup* and *MotherGroup*). It contains the columns of FatherMatches.csv and MotherMatches.csv with additional columns, as shown in Table 8.

Table 8 Columns in BothMatches.csv in addition to those in FatherMatches.csv and MotherMatches.csv.

Variable name	Description
mmrateF<fatherrank>M<motherrank>	The (raw) mismatch rate for possible parent matches, where *<fatherrank>* is 1 to indicate *BestFatherMatch* and 2 to indicate *FatherMatch2nd*, and similarly for *<motherrank>.*
mmnumF<fatherrank>M<motherrank>	The number of SNPs used to calculate *mmrateF<fatherrank>M<motherrank>*
exp.mmrateF<fatherrank>M<motherrank>	The expected mismatch rate corresponding to *mmrateF<fatherrank>M<motherrank>*
relF<fatherrank>M<motherrank>	The estimated relatedness between the pair of possible parents
Inb	The estimated inbreeding of the offspring
BothAssign	The code for the parent-pair assignment
Alternate	An alternative (to F1M1) parent pair

GroupsParentCounts.csv returns the groups file with additional columns, as shown in Table 9.
Table 9 Columns in GroupsParentCounts.csv in addition to those in the groups file.

Variable name	Description
FatherFreq	Number of offspring where this father is the *BestFatherMatch* in this group
MotherFreq	Number of offspring where this mother is the *BestMotherMatch* in this group

MatePairMatches.csv shows the results of father and mother pair matching (for individuals with *MatesGroup*). It contains the a subset of the columns of BothMatches.csv, given above (BothMatches.csv is not written with a mate pairs analysis). There are no *FatherAssign* or *MotherAssign* fields, no bootstrap results and no information parent pairs across the best and 2nd best matches (e.g. no F1M2 results). The fields occur in a different order to those in BothMatches.csv.

BestFatherMatches.png is a plot of the raw mismatch rate for *BestFatherMatch* against the estimated relatedness (*Fatherrel*). Points are coloured using *fcolo* and a grey vertical line indicates the value of *rel.thresh* used.

BestFatherMatchesE.png is the same BestFatherMatches.png except that the excess mismatch rate is plotted. A grey horizontal line indicates the value of *emm.thresh* used and grey shading indicates the excluded best matching fathers.

Best2FatherMatches.png is a plot of the estimated relatedness for *FatherMatch2nd* (*Fatherrel2nd*) against that for *BestFatherMatch* (*Fatherrel*). Points are coloured using a scale based on the excess mismatch rate (*mmrateFather- exp.mmrateFather*) for father-offspring and the line of equality is drawn (by definition all points fall below the line). Vertical and horizontal grey lines indicate the value of *rel.threshF* or *rel.threshM* (for fathers and mothers, respectively) used.

ExpMM-Father.png is a plot of the raw mismatch rate against the expected mismatch rate for *BestFatherMatch*. A red line shows where these are equal, and a grey line shows the boundary for an E assign code. Grey shading indicates the excluded best matching fathers. Points are coloured using *fcolo* and the symbols indicate *FatherAssign*.

BestMotherMatches.png, *BestMotherMatchesE.png*, *Best2MotherMatches.png* and *ExpMM-Mother.png* are the corresponding plots to BestFatherMatches.png, BestFatherMatchesE.png and Best2FatherMatches.png and ExpMM-Father.png, respectively, for mothers.

ParRel-Inb.png is a plot of offspring estimated inbreeding against estimated parent-pair relatedness (axes have been swapped from version 0). Points are coloured according to the mean depth in the offspring (as depth is more critical for inbreeding than relatedness estimation), and with a symbol corresponding to *BothAssign* (see ExpMM-Both.png for a key). Grey lines indicate the boundary for accepting parentage and the area below that line (excluded) is shaded grey.

MMrateBoth.png is a scatterplot matrix plot of the four combinations of parent-pair raw mismatch rates that were saved in BothMatches.csv. Points are coloured using *fcolo* and the lines of equality are drawn (in red).

MMrateBothE.png is a scatterplot matrix plot of the four combinations of parent-pair excess mismatch rates (or a single scatter plot of 2nd versus best EMM, when *matesfile* is used). Points are coloured using *fcolo*, the lines of equality are drawn (in red) and the symbols for the points denote *BothAssign*. The key for the symbols can be found in ExpMM-Both.png.

ExpMM-Both.png is a plot of raw versus expected parent-pair mismatch rates. A red line shows where these are equal, and a grey line shows the boundary for an E assign code. Grey shading indicates the excluded best matching parent pairs. Points are coloured using *fcolo* and the symbols for the points denote *BothAssign*.

Run a parentage analysis (*GBSPed*)

GBSPed is the function to run a parentage analysis.

Usage: GBSPed()
Details: Returns a list containing pedinfo (the pedigree file with additional information as output in PedVerify.csv), and one or more of FatherMatches, MotherMatches and BothMatches (which contain information about the parentage assignments, as written to FatherMatches.csv, MotherMatches.csv, and MatesMatches.csv or BothMatches.csv, respectively). If *functions.only* is FALSE, then sourcing GBSPedAssign.R will invoke the command:
PedResults <- GBSPed()
If some or all of the settings for pedigree analysis are not defined prior to use of *GBSPed*, they will be given their default values, if they are defined, and placed in the global environment.

Trio EMM sum of squares for beta-binomial model (*ssbbmm*)

A function to use for determining the fit of a beta-binomial model in terms of trio matches. Should only be run after a trio parentage assignment. *Under development and likely to change in future updates.*

Usage: ssbbmm(bbpar, ,uuse, BothMatches, quiet=FALSE)
Arguments:

bbpar	the parameter for the beta-binomial model
uuse	the offpring set to use (positions in BothMatches) as true trios
BothMatches	a data frame of parentage results, normally the *BothMatches* item in the list returned from *GBSPed*().
quiet	Set to TRUE to prevent parameter and EMM sum of squares being displayed. The default is FALSE.

Details: Returns the sum of squared trio EMM values for the uuse offspring and their assigned parents. The function can be used to estimate the beta-binomial parameter, e.g. with optimise(ssbbmm,lower=0,upper=20, tol=0.001)

Trio EMM sum of squares for modified p model (*ssmpmm*)

A function to use for determining the fit of a modified p model in terms of trio matches. Should only be run after a trio parentage assignment. *Under development and likely to change in future updates.*

Usage: ssmpmm(mppar, ,uuse=uY, BothMatches, quiet=FALSE)

Arguments:

mppar	the parameter for the modified p model
uuse	the offpring set to use (positions in BothMatches) as true trios
BothMatches	a data frame of parentage results, normally the *BothMatches* item in the list returned from *GBSPed*().
quiet	Set to TRUE to prevent parameter and EMM sum of squares being displayed. The default is FALSE.

Details: Returns the sum of squared trio EMM values for the uuse offspring and their assigned parents. The function can be used to estimate the modified p parameter, e.g. with optimise(ssmpmm, lower=0.5,upper=0.8, tol=0.001)

Add tagID function (*addtagIDs*)

This is a function will add alternative IDs for offspring, father and mother.

Usage: addtagIDs(sampinfo, indvar, tagvar, matchtype ="both", pedresults)

Arguments:

sampinfo	a dataframe containing the relevant IDs for all individuals
indvar	quoted text giving the name of the variable in sampinfo that corresponds to IndivID in the pedigree file
tagvar	quoted text giving the name of the variable in sampinfo that contains the tag (identifier) to be returned
matchtype	one of “both”, “father” or “mother” (not case-sensitive) specifying which data frame to update
pedresults	a data frame of pedigree assignment results from a pedigree analysis. Normally the *BothMatches, FatherMatches* or *MotherMatches* item in the list returned from *GBSPed*() (when *matchtype* is “both”, “father” or “mother”, respectively).

Details: Returns a data frame with the additional IDs added. It is not output to a file.

Parentage PC plot function (*bestparPCA*)

This is a function generates a PC2 vs PC1 plot with lines joining each progeny with its best father and mother.

Usage: bestparPCA(Gobj, sfx=””, keypos=NULL, pedinfo, BothMatches)

Arguments:

Gobj	an object produced from calcG (usually the same one used to obtain the GCheck matrix for parentage), with npc ≥ 2
sfx	text to be included in output file name to allow output from multiple calls or runs to be identified
keypos	the location (if given) on the plot of the legend of assign codes, using a value as accepted by the legend command (e.g. “topleft”)
pedinfo	a data frame of pedigree information, normally the *pedinfo* item in the list returned from *GBSPed*(). It must contain the columns *seqID* and *IndivID* as used in the pedigree analysis
BothMatches	a data frame of parentage results, normally the *BothMatches* item in the list returned from *GBSPed*().

Details: Generates a PC plot.

PC-BestParents<sfx>.png a plot of PC2 vcs PC1 relating to the parentage analysis. Points are coloured according to fcolo. Parents are shown as dots and offspring have symbols representing their assignment codes. Blue and pink lines are drawn from best father and best mother, respectively, to offspring.

Population genetics analysis (GBS-PopGen.R)

This R code makes available some functions for population genetics analyses. These are currently under development and only a brief description is provided here. The methods and syntax of this code is likely to change in the future.

Heterozygosity measures (*heterozygosity*)

This function gives various measures of observed and expected heterozygosity. keep.alleles should be set to TRUE to use this function.

Usage: heterozygosity(indsubsetgf=1:nind,snpsubsetgf=1:nsnps,maxiter=100,convtol=0.001)
Arguments:

indsubsetgf	a vector of integers (between 1 and *nind,* inclusive) of individuals to use for the heterozygosity measures. The default is to use all individuals.
snpsubsetgf	a vector of integers (between 1 and *nsnps,* inclusive) of SNPs to use for the heterozygosity measures. The default is to use all SNPs.
maxiter	maximum number of iterations to use in the estimation process. The default is 100.
convtol	convergence tolerance – the difference between genotype frequency estimates in successive iterations that is sufficiently small to assume convergence. The default value is 0.001.

Details: A data frame is returned with a (unlabelled) row for each SNP and the columns shown in Table 10.

Table 10 Columns in data frame returned from the heterozygosity function.

Variable name	Description
neff	Effective number of individuals (half the expected number of alleles in the data for a SNP, averaged over SNPs)
ohetstar	Observed heterozygosity on the raw scale
ehetstar	Observed heterozygosity on the raw scale (the proportion of genotype results expected to contain reads from both alleles)
ohet	Observed heterozygosity (estimated) on the true genotype scale
ohet2	An alternative measure of *ohet* (currently the preferred measure)
ehet	Expected heterozygosity (estimated) on the true genotype scale

F_{ST} calculations (*Fst.GBS* and *Fst.GBS.pairwise*)

These functions calculate approximate F_{ST} (estimates) accounting for read depth. The methods used can be found in Dodds *et al.*(2025).

Usage: Fst.GBS(snpsubset, indsubset, populations, varadj=0, SNPtest=FALSE) and/or Fst.GBS.pairwise (snpsubset, indsubset, populations,sortlevels=TRUE, SNPtest=FALSE, ...)
Arguments:

snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of SNPs to calculate F_{ST} for. The default is to use all SNPs.
indsubset	a vector of integers (between 1 and *nind,* inclusive) of individuals to use for the calculations. The default is to use all individuals.
populations	a vector of length *nind* containing population labels
varadj	use varadj=1 to get Fst as Weir p166, varadj=0 for usual Fst.

sortlevels	determines whether populations are listed as encountered in *populations* (sortlevels=FALSE) or sorted (sortlevels=TRUE)
SNPtest	if TRUE calculate and return p-values for each SNP. The default is FALSE. The type of output object will depend on SNPtest (see below).
…	arguments passed to Fst.GBS (currently only varadj)

Details: F_{ST} with depth adjustment. The adjustment is "approximate" and may result in estimates outside [0,1]. The .pairwise version calculates the statistics for each pair of populations. The (pairwise) means and medians are displayed. When SNPtest is FALSE (default) the values for each SNP are returned in a vector (Fst.GBS) or three-dimensional array (Fst.GBS.pairwise) with first two dimensions being the population and the third dimension being the SNP. When SNPtest is TRUE the output object is a list with first element, Fst, being the object that is returned when SNPtest is FALSE (i.e., a vector or three-dimensional array) and the second element, pvalue, is the same dimensional object of p-values (corresponding to the Fst values).

Genomic relatedness by population (*popG*)

This function computes the average of a G matrix by populations (without self-reatednessl) and the mean inbreeding by population.

Usage: popG(Guse, populations, diag=FALSE)

Arguments:

Guse	the GRM for all individuals.
populations	a vector containing population labels. This should be the same length as the dimensions of the GRM.
diag	Should the diagonal elements include individual self-relatedness values. The default is FALSE in which case the diagonal elements of the GRM returned are mean relatedness between different individuals in the corresponding population.

Details: The output object is a list containing G and Inb. G is the genomic relatedness between populations (having a row and column for each population).

MAF plots by population (*popmaf*)

Plot minor allele frequency distributions by population.

Usage: popmaf(snpsubset, indsubset, populations=NULL, subpopulations=NULL, indcol, colobj, minsamps=10, mafmin=0, sortlevels=TRUE, unif=FALSE)

Arguments:

snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of SNPs calculate F_{ST} for. The default is to use all SNPs.
indsubset	a vector of integers (between 1 and *nind,* inclusive) of individuals to use for the calculations. The default is to use all individuals.
populations	a vector of length *nind* containing population labels. The default is NULL in which case a MAF plot for all indsubset individuals is given.
subpopulations	a vector of length *nind* containing subpopulation labels. These are treated as being nested within *populations*.
indcol	A colour assignment for each individual (length *nind*). The plot is coloured by *indcol* if all individuals in that population have that colour (otherwise black).
colobj	an object created by *colourby* based on *nind* individuals. If present it will be used for populations and their colours.
minsamps	Minimum number of samples in a (sub)population to invoke plotting. The default is 10.
mafmin	Minimum (sub)population MAF to include a SNP in the plot. The default is 0.
sortlevels	determines whether data are processed by populations as encountered (sortlevels=FALSE) or sorted (sortlevels=TRUE)
unif	determines whether the plots are drawn with the same vertical axis range. The default is FALSE (population-specific range).

Details: A MAF distribution is plotted, possibly with a different colour for each population, and different shading for each subpopulation. Some summary statistics are also displayed.

Discriminant analysis of principal components (*DAPC.GBS*)

This function produces a discriminant analysis of principal components (DAPC; Jombart *et al.* 2010) with some basic settings. The function uses a set of predefined groups (populations); as yet KGD does not include a specific function for grouping individuals. The output can be visualised passing it into *GRMPCA* with the PCobj argument. The MASS package must be installed to use this function.

Usage: DAPC.GBS(Guse, populations=NULL, n.pca=NULL, perc.pca=90)

Guse	The GRM to use for PCA, which is then the input for DAPC. This argument must be given.
populations	a grouping variable. The values correspond to the rows (& columns) of *Guse*. This argument must be given.
n.pca	the number of principal components to use in the linear discriminant analysis. function name of null distribution. If not given, the number is determined from the *perc.pca* argument.
perc.pca	used to calculate *n.pca* when that is not given. The number of principal components will be that required to explain at least *perc.pca*% of the variation. The default value is 90.

Details: The output object is a list as returned by the *lda* function of MASS, with an additional element *x*, containing the values on the discriminant axes.

Manhattan plots (*manhatplot*)

Simple plotting of results as a Manhattan plot.

Usage: manhatplot(value, chrom, pos, plotname, qdistn=qunif, keyrot=0, symsize=0.8, legendm = NULL, ...)

value	vector of statistic values to be plotted (for each SNP).
chrom	chromosome name (numeric or character) for each SNP.
pos	position (numeric) on chromosome for each SNP.
plotname	text used as prefix for names of output plots
qdistn	function name of null distribution. The default is the uniform distribution (qunif) which could be used e.g., with p-values. A more common example would be the chi-squared distribution (qchisq).
keyrot	rotation angle for chromosome key. The default is 0, but 90 is a better choice for longer chromosome labels.
symsize	the size of the symbols that are plotted. If this is a vector of the same length as *value* they are the sizes of each point plotted. The default is 0.8 (for all points).
legendm	a function executed after each plot (normally a function to place a legend on the plots).
…	further arguments passed to qdistn, for example the df (degreeso of freedom) parameter (if using qdistn=qchisq).

Details: Two plots are produced. A text version of *value* is used in the y-axis labels, but with non-alphanumeric values replaced with "." (to prevent R interpreting them as instructions).

<plotname>-Manhat.png a Manhattan plot of *value.* Points are sorted by *chrom* and *pos* with a different colour for each value of *chrom*.

<plotname>-QQ.png a QQ plot *value* using the distribution specified in *qdistn*.

Pairs of SNPs from different chromosomes (*snpselection*, *snpselectionUR*)

These functions allow pairs of SNPs from different chromosomes to be selected, normally for use in calculating linkage disequilibrium between unlinked SNPs to allow effective population size estimates.

Usage: snpselection (chromosome, position, nsnpperchrom=100, seltype="centre", randseed=NULL, snpsubset, chromuse)

chromosome	a character vector of the chromosome name for each SNP
position	numeric vector of position (normally in bp) on chromosome for each SNP
nsnpperchrom	the (maximum) number of SNPs to choose on each chromosome. The default is 100.
seltype	the method to select SNPs form those available on a chromosome. Available values are "centre" – choose the *nsnpperchrom* SNPs closest to the centre (mean SNP position) of the chromosome; "even" – space the SNPs as evenly as possible (by SNP order on the chromosome, not using position values); or "random" – randomly selected
randseed	a seed value for the random number generator (to allow reproducible selection using "random" *seltype*)
snpsubset	a vector of integers (between 1 and *nsnps,* inclusive) of SNPs to choose from
chromuse	a character vector of the chromosomes (names, as in *chromosome*) to use. The default is to use all chromosomes. For example, this option allows restricting the selection of SNPs to the mapped autosomal SNPs.

Details: the output is an array with two columns of SNP positions, containing all pairs of selected SNPs where the pair are on different chromosomes.

Usage: snpselectionUR (URobj, nsnpperchrom=100, nchrom, …)

URobj	a UR (unrelated samples) object from the GUSbase package.
nsnpperchrom	the (maximum) number of SNPs to choose on each chromosome. The default is 100.
nchrom	the number of chromosomes to use. If specified, the first *nchrom* chromosomes are used. The default is to use all chromosomes.
…	additional parameters to be passed to *snpselection*.

Details: provides convenient access to the *snpseleciton* function for a UR object in GUSbase.

Effective population size (*Nefromr2*)

The *Nefromr2* function is experimental may be enhanced and/or modified at a later date. Calculates effective population size (N_e) from a set of linkage disequilibrium r^2 values. Assumes that these are from unlinked pairs of SNPs, and that the N_e is for the generation of the genotyped individuals (i.e. present day, rather than historic). The GUS-LD package (Bilton *et al.*, 2018) provides a method to estimate linkage disequilibrium while accounting for read depth and sequencing error.

Usage: Nefromr2 (r2auto, nLD, alpha=1, weighted=FALSE, minN=1)

r2auto	a vector of linkage disequilibrium r^2 values UR (unlinked autosomal pairs of SNPs).
nLD	the number of individuals used to calculate the r^2 values. Either a vector of the same length as *r2auto* or a single value (assumed to be the same for all pairs of SNPs).
alpha	the α (mutation) parameter. The default value is 1.
weighted	a setting to invoke using a weighted mean of r^2 values (when TRUE) in the calculations. The default is FALSE.
minN	a minimum number of individuals used to retain a pair of SNPs. The default value is 1 (i.e., no minimum).

Details: a set of statistics is displayed:

n	the mean sample size for a pair of SNPs

Neauto	the estimated Ne based on mean r^2, without bias correction
Neauto.adj.b1	the estimated Ne with bias correction using β=1 (for phase unknown data)
Neauto.adj.b2	the estimated Ne with bias correction using β=2 (for phase known data)
Neauto.med	the estimated Ne based on the median r^2 (without bias correction)
Neauto.med.adj.b1	the estimated Ne using medians, with ‘bias correction’ using β =1. For illustration only. Do not use.
Neauto.med.adj.b2	the estimated Ne using medians, with ‘bias correction’ using β =2. For illustration only. Do not use.

See Barbato *et al.* (2015) for a description and definition of α and β.

Input formats

The genotype input format is set with *gform*, one of “uneak” (the default), “Tassel”, “TagDigger” or “Chip”. Quotes in the input files will be treated as characters (not interpreted as quoting text).

GBS via UNEAK

The default input format (‘uneak’) is a ‘hapmap count’ formatted file as produced by the UNEAK pipeline (Lu *et al*. 2013). This is a tab-separated flat text file with the first column being the SNP identifier, then a column for each genotyped individual (or sample, or other genotyping unit), followed by 5 columns of summary information (HetCount_allele1, HetCount_allele2, Count_allele1, Count_allele2, Frequency). Only the last of these 5 is used. Each row is for a different SNP. The column for each individual contains the genotype information as the allele depth (number of reads of that allele) for the ‘reference’ and ‘alternate’ alleles, respectively. The designation of reference and alternate is arbitrary for this software. The numbers of reads are separated by a pipe symbol (“|”). There is a header line, which, for the genotype columns, is taken as the identifiers of the individuals.

GBS via Tassel

An additional format (‘Tassel’) is available that may be easier to use for GBS data that has been manipulated in Tassel. It is similar to the uneak format, but allele depths in a genotype are separated by a comma (“,”), has two columns before genotype data (, and no columns following the genotype data. The first two columns are the chromosome and position (which together, separated by an underscore, serve as the SNP identifier), respectively. As with the “uneak” format, this is a tab- separated flat text file with a header row.

GBS via TagDigger

TagDigger (https://github.com/lvclark/tagdigger, Clarke and Sacks, 2016) is a tool for SNP calling from a given set of tags (sequences). It is likely to be used in a production environment, where the set of SNPs being called is unlikely to change much with additional samples being added. The ‘TagDigger‘ format requires a comma delimited file with sample results in rows and SNP results in pairs of columns (count of reference allele, count of alternate allele). The first column contains the sample identifier. The header row, apart from the first value, contains SNP/allele identifiers. It is assumed that these identifiers have a SNP identifier followed by an underscore, followed by the allele identifier. The text preceding the underscore is taken as the SNP name (the other text is ignored).

TagDigger files will be read with the fread function from the data.table package, if that package is installed. This is faster than the method used when the package is not available. Files compressed with the gzip (.gz) format can be read by both methods, but may require the R.utils package if using fread (depending on the package versions).

GBS via ANGSD

ANGSD (http://www.popgen.dk/angsd, Korneliussen *et al*., 2014) is a program for analysing sequencing data, and can output SNP information. The 'ANGSDcount' format reads files created by the –dumpCounts 4 option of ANGSD. This file has a header row, followed by a row for each SNP. There is a column for each of the 4 possible alleles (A, C, G, T) for each SNP and sample. The columns for a sample are together. The header contains an identifier for each column consisting of the sample identifier followed by underscore and the allele (e.g. ind0_A). After reading this file, SNPs are checked for which alleles are most common. The two most common alleles are taken as the variant of interest, and other alleles are ignored, except that a SNP is discarded if the proportion of reads for the third most common alleles exceeds the threshold triallelic.thresh. SNPs are named as 'SNP' followed by the zero-padded position. SNPs that have been dropped by the triallelic threshold can be identified by finding gaps in the SNP_Name sequence.

vcf files

VCF files with GT and/or AD fields can be read with read.vcf.

A python helper script vcf2ra.py is also available to convert a wider range of .vcf files (named <infile>.vcf) to the 'Tassel' format (named <infile>.vcf.ra.tab). The .vcf file must have either the AD (allelic depth) field, or both the AO (alternate allele observation count) and RO (reference allele observation count) fields. Some sites are filtered out and placed in ancillary output files:

- indels are removed and reported in <infile>.vcf.indel,
- SNPs that are more than biallelic are removed and reported in <infile>.vcf.polyallele
- redundant sites are removed and reported in <infile>.vcf.posred

Chip

Fully recorded genotypes can be entered via the "Chip" format. This comma-separated format has results for each individual in the rows and SNP results in a column. There is a header row (SNP identifiers) and the first column contains individual identifiers. Subsequent columns contain the SNP results. Genotype data is given in 0/1/2 format, representing first homozygote, heterozygote and second homozygote, respectively. Designation of which allele is the 'first' is arbitrary. The second homozygote is taken to be homozygous at the reference allele (the allele for which allele frequencies are calculated). Genotypes outside the range [0,2] are treated as missing.

Chip format data can either be read with *readGBS*, or directly with *readChip*. The latter allows a different separator (sep parameter) and can have a missing column heading for the first (identifier) column (using row.names=FALSE).

Pedigree file

An optional pedigree file (given by *pedfile*) can be given and will be used to verify or find parent matches. This is a comma separated file (csv). All individuals to be considered as offspring or parents need to have a row in this file. The columns of this file are specified in Table 11. The names must be exactly as specified. Additional columns may be present in the file.

Table 11 Columns in the pedigree file that are used in the pedigree analysis, when relevant.

Variable name	Required?	Description
IndivID	Y	identifies individuals in the pedigree and groups files
seqID	Y	matches *IndivID* to the identifier in the genotype file
FatherID	N	Recorded *IndivID* of father
MotherID	N	Recorded *IndivID* of mother
FatherGroup	N	Group label for group of potential fathers for the given *IndivID*
MotherGroup	N	Group label for group of potential mothers for the given *IndivID*

MatesGroup	N	Group label for group of potential parent (mate) pairs for the given *IndivID*.

Father and mother group labels should be distinct. If required, they are entered for the progeny. The information linking these labels to the set of possible parents is placed in the groups file. Similarly, if *matesfile* is given, the *MatesGroup* field needs to give the label to identify possible parent pairs in the mates file.

Groups file

If parent matching is required, then a groups file (or a mates file, see below) (given by *groupsfile*) describing the group labels in the pedigree file is required. This is a comma separated file (csv). The columns (both required) of this file are specified in Table 12. The names must be exactly as specified. Additional columns may be present in the file.

Table 12 Columns in the groups file that are used in the pedigree analysis.

Variable name	Description
IndivID	identifier for potential parent, matching *IndivID* in the pedigree file
ParGroup	Group label for the group that *IndivID* belongs to

There should be one row for each group a potential parent belongs to. A warning is issued if a *ParGroup* (either *MotherGroup* or *FatherGroup*) is specified in the pedigree file but has no genotyped individuals.

Mates file

When parent matching is required and the mating pairs are known, then a mates file (given by *matesfile*) describing the mate group labels in the pedigree file is required. This is a comma separated file (csv). The columns (all required) of this file are specified in Table 13. The names must be exactly as specified. Additional columns may be present in the file.

Table 13 Columns in the mates file that are used in the pedigree analysis.

Variable name	Description
MaleID	identifier for potential male parent, matching *IndivID* in the pedigree file
FemaleID	identifier for potential female parent, matching *IndivID* in the pedigree file
MatesGroup	Group label for the group of mate pairs that could be the parent of *IndivID*

There should be one row for each group a potential parent belongs to.

Associated packages

The software has been designed to (mainly) run without the need for any R packages to be installed but can use such packages if available. Sometimes there will be messages relating to these packages, but these messages can be ignored. A list of optional and required packages and their use is given in Table 14.

Table 14 Optional and required packages that may be used with the KGD software.

Package name	Usage
data.table	Reading tagdigger and possibly Tassel input format files, writing VCF files, **required** for reading VCF files, **required** for writing gzipped G matrices (writeG).
heatmaply	Interactive heatmap from calcG
irlba	Used for PCA when use.irlba is TRUE in calcG or GRMPCA.
parallelDist	Parallelized calculation of distance for the heatmap in calcG

plotly	Interactive graphics output from calcG
MASS	**Required** for DAPC.GBS. MASS is normally shipped with R.
MethComp	Bland-Altman plots in GCompare
Rcpp	Various functions have C++ versions for improved efficiency, but require this package
RcppArmadillo	This package is required for some of the C++ functions to be used
R.utils	**required** for reading VCF files (if required by the data.table version)
Irlba	Used for fast PCA calculations if available and this method is requested.

Some of the structures defined in the KGD software may be incompatible with those defined in some packages. A known issue is that adegenet defines an *alleles* function while KGD defines an *alleles* matrix. Therefore adegenet should not be loaded until at least after using KGD to read the datafile.

Example

This folder contains an example run (possibly using an earlier version). Files in directory :
GBSRun.R HapMap.hmc.txt.gz Ped-GBS.csv Ped-Groups.csv

GBSRun.R
```
genofile <- "HapMap.hmc.txt.gz"

source("<source directory>/GBS-Chip-Gmatrix.R")
Gfull <- calcG()
GHWdgm.05 <- calcG(which(HWdis > -0.05),"HWdgm.05", npc=4)  # recalculate
using Hardy-Weinberg disequilibrium cut-off at -0.05

pedfile <- "Ped-GBS.csv"
groupsfile <- "Ped-Groups.csv"

rel.thresh <- 0.2
emm.thresh <- 0.075  # to make results same as before emm used (to match
original example)
GCheck <- "GHWdgm.05$G5"
source("<source directory>/GBSPedAssign.R")
```

<source directory> should be replaced with the location of the relevant .R files before running.
linux command:
R CMD BATCH --no-save GBSRun.R &

Files in directory after running code:
```
AlleleFreq.png             GcompareHWdgm.05.png        PC1v2G5HWdgm.05.png
Best2FatherMatches.png     Gcompare.png                PC1vDepthHWdgm.05.png
Best2MotherMatches.png     Gdiagdepth.png              PC1vInbHWdgm.05.png
BestFatherMatchesE.png     G-diag.png                  PCG5HWdgm.05.pdf
BestFatherMatches.png      GHWdgm.05diagdepth.png      Ped-GBS.csv
BestMotherMatchesE.png     GHWdgm.05-diag.png          Ped-Groups.csv
BestMotherMatches.png      GroupsParentCounts.csv      PedVerify.csv
BothMatches.csv            HapMap.hmc.txt.gz           SampDepthCR.png
CallRate.png               Heatmap-G5HWdgm.05.png      SampDepthHist.png
Co-call-HWdgm.05.png       HeatmapOrderHWdgm.05.csv    SampDepth.png
Co-call-.png               HighRelatednessHWdgm.05.csv SampDepth-scored.png
ExpMM-Father.png           HWdisMAFsig.png             SampleStats.csv
ExpMM-Mother.png           LRT-hist.png                seqID.csv
FatherMatches.csv          LRT-QQ.png                  SNPCallRate.png
FatherVerify.png           MAFHWdgm.05.png             SNPDepthHist.png
finplot.png                MAF.png                     SNPDepth.png
GBSRun.R                   MotherMatches.csv           X2star-QQ.png
GBSRun.Rout                MotherVerify.png
```

A workshop using this example was given at the 2015 MapNet meeting (Rotorua, New Zealand). Instructions (KGDCourseInstructions-Mapnet2015.pdf) and course notes (KGDCourse-Mapnet2015.pdf) are available in the Example folder.

ParExample

This folder gives an example of the code for a parentage analysis, based on the example given in Dodds *et al*. (2019). Example code is given in GBSParDeer.R. The example code assumes all necessary files are in the working directory. The example data (allele counts, pedigree file, groups file) can be obtained from https://gsajournals.figshare.com/s/7ca45accf6ae82047c86. An annotated description of the commands in GBSParDeer.R is in GBSParentage-Annotated.pdf.

PopGExamples

This folder contains FstSim.R which regenerates the simulation data from Dodds *et al.*(2025).

Acknowledgement

This project was supported by the Ministry of Business, Innovation and Employment via its funding of the "Genomics for Production & Security in a Biological Economy" programme (Contract ID C10X1306).

Rudi Brauning, Rachael Ashby, Timothy Bilton, Alice Chappell and David Winter have all contributed to the code development.

References

Altman, D G and Bland, J M (1983) Measurement in medicine: the analysis of method comparison studies. *The Statistician* **32**, 307-337.

Auvray, B, McEwan, J C, Newman, S A N, Lee, M and Dodds, K G (2014) Genomic prediction of breeding values in the New Zealand sheep industry using a 50K SNP chip. *Journal of Animal Science* **92**, 4375-4389. doi:10.2527/jas.2014-7801

Barbato, M, Orozco-terWengel, P, Tapio, M and Bruford, M W (2015) *SNeP*: a tool to estimate trends in recent effective population size trajectories using genome-wide SNP data. *Frontiers in Genetics* **6**, doi:10.3389/fgene.2015.00109

Bilton, T P, McEwan, J C, Clarke, S M, Brauning, R, Van Stijn, T C, Rowe, S J and Dodds, K G (2018) Linkage disequilibrium estimation in low coverage high-throughput sequencing data. *Genetics* **209**, 389-400. doi:10.1534/genetics.118.300831

Bilton, T P, Chappell, A J, Clarke, S M, Brauning, R, Dodds, K G, McEwan, J C and Rowe, S J (2019) Using genotyping-by-sequencing to predict gender in animals. *Animal Genetics* **50**, 307-310. doi:10.1111/age.12782

Carstensen, B, Gurrin, L, Ekstrom, C and Figurski, M (2015). MethComp: Functions for Analysis of Agreement in Method Comparison Studies. R package version 1.22.2. http://CRAN.R-project.org/package=MethComp

Cericola, F, Lenk, I, Fè, D, Byrne, S, Jensen, C, Pedersen, M, Asp, T, Jensen, J and Janss, L (2018) Optimized use of low-depth genotyping-by-sequencing for genomic prediction among multi-parental family pools and single plants in perennial ryegrass (*Lolium perenne* L.). *Frontiers in Plant Science* **9**, 369. doi:10.3389/fpls.2018.00369

Clark, L V and Sacks, E J (2016) TagDigger: user-friendly extraction of read counts from GBS and RAD-seq data. *Source Code for Biology and Medicine* **11**, 1-6. doi:10.1186/s13029-016-0057-7

Dodds, K G, McEwan, J C, Brauning, R, Anderson, R A, Van Stijn, T C, Kristjánsson, T and Clarke, S M (2015) Construction of relatedness matrices using genotyping-by-sequencing data. *BMC Genomics* **16**, 1047.

Dodds, K G, McEwan, J C, Brauning, R and Clarke, S M (2025) Assessing population allele frequency differences using low-depth sequencing data. *Journal of the Royal Society of New Zealand* **55**, 2677–2688. doi:10.1080/03036758.2025.2500999

Dodds, K G, McEwan, J C, Brauning, R, Van Stijn, T C, Rowe, S J, McEwan, K M and Clarke, S M (2019) Exclusion and genomic relatedness methods for assignment of parentage using genotyping-by-sequencing data. *G3: Genes, Genomes, Genetics* **9**, 3239-3247. doi: 10.1534/g3.119.400501.

Harris, B L and Johnson, D L (2010) Genomic predictions for New Zealand dairy bulls and integration with national genetic evaluation. *Journal of Dairy Science* **93**, 1243-1252.

Jombart, T, Devillard, S and Balloux, F (2010) Discriminant analysis of principal components: A new method for the analysis of genetically structured populations. *BMC Genetics* **11**, 94.

Korneliussen, T S, Albrechtsen, A and Nielsen, R (2014) ANGSD: Analysis of Next Generation Sequencing Data. *BMC Bioinformatics* **15**, 356. doi:10.1186/s12859-014-0356-4

Kuehn, L A, Keele, J W, Bennett, G L, McDaneld, T G, Smith, T P L, Snelling, W M, Sonstegard, T S and Thallman, R M (2011) Predicting breed composition using breed frequencies of 50,000 markers from the US Meat Animal Research Center 2,000 Bull Project. *Journal of Animal Science* **89**, 1742–1750. doi:10.2527/jas.2010-3530

Li, H (2011) A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics* **27**, 2987-2993.

Lu, F, Lipka, A E, Glaubitz, J, Elshire, R, Cherney, J H, Casler, M D, Buckler, E S and Costich, D E (2013) Switchgrass Genomic Diversity, Ploidy, and Evolution: Novel Insights from a Network-Based SNP Discovery Protocol. *PLoS Genetics* **9**, e1003215.

McKinney, G J, Waples, R K, Seeb, L W and Seeb, J E (2017) Paralogs are revealed by proportion of heterozygotes and deviations in read ratios in genotyping-by-sequencing data from natural populations. *Molecular Ecology Resources* **17**, 656-669. doi:10.1111/1755-0998.12613

Schaeffer, L (2013) Making covariance matrices positive definite. http://animalbiosciences.uoguelph.ca/~lrs/ELARES/PDforce.pdf

Reverter, A, Porto-Neto, L R, Fortes, M R S, McCulloch, R, Lyons, R E, Moore, S, Nicol, D, Henshall, J and Lehnert, S A (2016) Genomic analyses of tropical beef cattle fertility based on genotyping pools of Brahman cows with unknown pedigree. *Journal of Animal Science* **94**, 4096-4108. doi:10.2527/jas.2016-0675

Weir, B S and Goudet, J (2017) A unified characterization of population structure and relatedness. *Genetics* **206**, 2085-2103. doi:10.1534/genetics.116.198424