Package 'polyRAD'

June 4, 2019

Version 1.1

Date 2019-06-04

Title Genotype Calling with Uncertainty from Sequencing Data in Polyploids and Diploids

Author Lindsay V. Clark [aut, cre] (<https://orcid.org/0000-0002-3881-9252>), U.S. National Science Foundation [fnd]

Maintainer Lindsay V. Clark <lvclark@illinois.edu>

Imports fastmatch, pcaMethods, methods, Rcpp

Suggests rrBLUP, Rsamtools, GenomeInfoDb, Biostrings, GenomicRanges, VariantAnnotation, SummarizedExperiment, S4Vectors, IRanges, BiocGenerics, knitr, rmarkdown, GenomicFeatures, qqman

LinkingTo Rcpp

VignetteBuilder knitr, rmarkdown

Description Read depth data from genotyping-by-sequencing (GBS) or restriction site-associated DNA sequencing (RAD-seq) are imported and used to make Bayesian probability estimates of genotypes in polyploids or diploids. The genotype probabilities, or genotypes sampled from those probabilties, can then be exported for downstream analysis. 'polyRAD' is described by Clark et al. (2019) <doi:10.1534/g3.118.200913>.

License GPL (>= 2)

URL https://github.com/lvclark/polyRAD

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-06-04 20:00:03 UTC

R topics documented:

`Accessors` *Accessor Functions for RADdata Objects*

Description

These functions can be used for accessing and replacing data within a "`RADdata`" object. Data slots that do not yet have accessors can be accessed and replaced using the `$` operator or the `attr` function.

Usage

```
GetTaxa(object, ...)
GetLoci(object, ...)
GetLocDepth(object, ...)
GetContamRate(object, ...)
SetContamRate(object, value, ...)
nTaxa(object, ...)
nLoci(object, ...)
nAlleles(object, ...)
GetAlleleNames(object, ...)
```

Arguments

object	A "RADdata" object.
value	A value to assign. For SetContamRate, a number generally ranging from zero to 0.01 indicating the expected rate of sample cross-contamination.
...	Additional arguments (none currently supported).

Value

For GetTaxa and GetLoci, a character vector listing taxa names or loci names, respectively. For GetLocDepth, a named matrix with taxa in rows and loci in columns, giving the total read depth for each taxon and locus. For GetContamRate, a number indicating the expected contamination rate that is stored in the object. For SetContamRate, a "RADdata" object with an updated contamination rate. For nTaxa, the number of taxa in the object. For nLoci, the number of loci in the object. For nAlleles, the number of alleles across all loci in the object. For GetAlleleNames, the names of all alleles.

Author(s)

Lindsay V. Clark

See Also

SetBlankTaxa for functions that assign taxa to particular roles.

Examples

```
data(exampleRAD)
GetTaxa(exampleRAD)
GetLoci(exampleRAD)
GetLocDepth(exampleRAD)
GetContamRate(exampleRAD)
exampleRAD <- SetContamRate(exampleRAD, 0.0000001)
GetContamRate(exampleRAD)
nTaxa(exampleRAD)
nAlleles(exampleRAD)
GetAlleleNames(exampleRAD)
```

AddAlleleFreqByTaxa *Estimate Local Allele Frequencies for Each Taxon Based on Population Structure*

Description

This function estimates allele frequencies per taxon, rather than for the whole population. The best estimated genotypes (either `object$depthRatio` or `GetWeightedMeanGenotypes(object)`) are regressed against principal coordinate axes. The regression coefficients are then in turn used to predict allele frequencies from PC axes. Allele frequencies outside of a user-defined range are then adjusted so that they fall within that range.

Usage

```
AddAlleleFreqByTaxa(object, ...)
## S3 method for class 'RADdata'
AddAlleleFreqByTaxa(object, minfreq = 0.0001, ...)
```

Arguments

`object`	A "RADdata" object. AddPCA should have already been run.
`minfreq`	The minimum allowable allele frequency to be output. The maximum allowable allele frequency will be calculated as `1 - minfreq`.
`...`	Additional arguments (none implemented).

Details

For every allele, all PC axes stored in `object$PCA` are used for generating regression coefficients and making predictions, regardless of whether they are significantly associated with the allele.

`object$depthRatio` has missing data for loci with no reads; these missing data are omitted on a per-allele basis when calculating regression coefficients. However, allele frequencies are output for all taxa at all alleles, because there are no missing data in the PC axes. The output of GetWeightedMeanGenotypes has no missing data, so missing data are not an issue when calculating regression coefficients using that method.

After predicting allele frequencies from the regression coefficients, the function loops through all loci and taxa to adjust allele frequencies if necessary. This is needed because otherwise some allele frequencies will be below zero or above one (typically in subpopulations where alleles are near fixation), which interferes with prior genotype probability estimation. For a given taxon and locus, any allele frequencies below `minfreq` are adjusted to be equal to `minfreq`, and any allele frequencies above `1 - minfreq` are adjusted to be `1 - minfreq`. Remaining allele frequencies are adjusted so that all allele frequencies for the taxon and locus sum to one.

Value

A "RADdata" object identical to the one passed to the function, but with a matrix of allele frequencies added to the `$alleleFreqByTaxa` slot. Taxa are in rows and alleles in columns.

Author(s)

Lindsay V. Clark

See Also

AddGenotypePriorProb_ByTaxa

Examples

```
# load data
data(exampleRAD)
# do PCA
exampleRAD <- AddPCA(exampleRAD, nPcsInit = 3)

# get allele frequencies
exampleRAD <- AddAlleleFreqByTaxa(exampleRAD)

exampleRAD$alleleFreqByTaxa[1:10,]
```

AddAlleleFreqHWE *Estimate Allele Frequencies in a RADdata Object Assuming Hardy-Weinberg Equilibrium*

Description

Allele frequencies are estimated based on the best parameters available. `object$alleleFreqByTaxa` is used if available. If `object$alleleFreqByTaxa` is null, `GetWeightedMeanGenotypes` is used, and if that isn't possible `object$depthRatio` is used. From whichever of the three options is used, column means are taken, the output of which is stored as `object$alleleFreq`.

Usage

```
AddAlleleFreqHWE(object, ...)
## S3 method for class 'RADdata'
AddAlleleFreqHWE(object, excludeTaxa = GetBlankTaxa(object), ...)
```

Arguments

`object`	A `"RADdata"` object.
`excludeTaxa`	A character vector indicating taxa that should be excluded from the calculation.
`...`	Included to allow more arguments in the future, although none are currently used.

Value

A `"RADdata"` object identical to the one passed to the function, but with allele frequencies added to `object$alleleFreq`, and `"HWE"` as the value for the `"alleleFreqType"` attribute.

Author(s)

Lindsay V. Clark

See Also

AddAlleleFreqMapping, AddGenotypePriorProb_HWE

Examples

```
# load in an example dataset
data(exampleRAD)
exampleRAD

# add allele frequencies
exampleRAD <- AddAlleleFreqHWE(exampleRAD)
exampleRAD$alleleFreq
```

AddAlleleFreqMapping *Estimate Allele Frequencies in a Mapping Population*

Description

Estimate allele frequencies using data from a mapping population, assuming a fixed set of allele frequencies are possible.

Usage

```
AddAlleleFreqMapping(object, ...)
## S3 method for class 'RADdata'
AddAlleleFreqMapping(object, expectedFreqs = seq(0, 1, 0.25),
                     allowedDeviation = 0.05,
                     excludeTaxa = c(GetDonorParent(object),
                                     GetRecurrentParent(object),
                                     GetBlankTaxa(object)), ...)
```

Arguments

object	A "RADdata" object. The donor and recurrent parent should have been assigned with SetDonorParent and SetRecurrentParent, respectively. If this is not a backcross population, it does not matter which is the donor or recurrent parent.
expectedFreqs	A numeric vector listing all expected allele frequencies in the mapping population.
allowedDeviation	A value indicating how far an observed allele frequency can deviate from an expected allele frequency and still be categorized as that allele frequency. Must be no more than half the smallest interval seen in expectedFreqs.
excludeTaxa	A character vector indicating taxa that should be excluded from the allele frequency estimate.
...	Arguments to be passed to the method for "RADdata".

Details

Allele frequencies are first estimated as the column means of `object$depthRatio` (unless posterior genotype probabilities and ploidy chi-squared values have already been calculated, in which case `GetWeightedMeanGenotypes` is run and the column means of its output are taken), excluding any taxa listed in `excludeTaxa`. These are then categorized based on which, if any, expected allele frequency they match with, based on the intervals described by `expectedFreqs` and `allowedDeviation`. If an allele frequency does not fall within any of these intervals it is classified as `NA`; otherwise it is converted to the matching value in `expectedFreqs`.

Value

A `"RADdata"` object identical to the one passed to the function, but with allele frequencies added to `object$alleleFreq`, and `"mapping"` as the `"alleleFreqType"` attribute.

Author(s)

Lindsay V. Clark

See Also

`AddAlleleFreqHWE`

Examples

```
# load example dataset
data(exampleRAD_mapping)
exampleRAD_mapping

# specify parents
exampleRAD_mapping <- SetDonorParent(exampleRAD_mapping, "parent1")
exampleRAD_mapping <- SetRecurrentParent(exampleRAD_mapping, "parent2")

# estimate allele frequencies in diploid BC1 population
exampleRAD_mapping <- AddAlleleFreqMapping(exampleRAD_mapping,
                        expectedFreqs = c(0.25, 0.75),
                        allowedDeviation = 0.08)

exampleRAD_mapping$alleleFreq
```

AddAlleleLinkages *Identify and Utilize Linked Alleles for Estimating Genotype Priors*

Description

`AddAlleleLinkages` finds alleles, if any, in linkage disequilibrium with each allele in a `RADdata` object, and computes a correlation coefficient representing the strength of the linkage. `AddGenotypePriorProb_LD` adds a second set of prior genotype probabilities to a `RADdata` object based on the genotype posterior probabilities at linked alleles.

Usage

```
AddAlleleLinkages(object, ...)
## S3 method for class 'RADdata'
AddAlleleLinkages(object, type, linkageDist, minCorr,
                  excludeTaxa = character(0), ...)

AddGenotypePriorProb_LD(object, ...)
## S3 method for class 'RADdata'
AddGenotypePriorProb_LD(object, type, ...)
```

Arguments

`object`	A `RADdata` object with genomic alignment data stored in `object$locTable$Chr` and `object$locTable$pos`.
`type`	A character string, either "mapping", "hwe", or "popstruct", to indicate the type of population being analyzed.
`linkageDist`	A number, indicating the distance in basepairs from a locus within which to search for linked alleles.
`minCorr`	A number ranging from zero to one indicating the minimum correlation needed for an allele to be used for genotype prediction at another allele.
`excludeTaxa`	A character vector listing taxa to be excluded from correlation estimates.
`...`	Additional arguments (none implemented).

Details

These functions are primarily designed to be used internally by the pipeline functions.

`AddAlleleLinkages` obtains genotypic values using `GetWeightedMeanGenotypes`, then regresses those values for a given allele against those values for nearby alleles to obtain correlation coefficients. For the population structure model, the genotypic values for an allele are first regressed on the PC axes from `object$PCA`, then the residuals are regressed on the genotypic values at nearby alleles to obtain correlation coefficients.

`AddGenotypePriorProb_LD` makes a second set of priors in addition to `object$priorProb`. This second set of priors has one value per inheritance mode per taxon per allele per possible allele copy number. Where K is the ploidy, with allele copy number c ranging from 0 to K, i is an allele, j is a linked allele at a different locus out of J total alleles linked to i, r_{ij} is the correlation coefficient between those alleles, t is a taxon, $post_{cjt}$ is the posterior probability of a given allele copy number for a given allele in a given taxon, and $prior_{cit}$ is the prior probability for a given allele copy number for a given allele in a given taxon based on linkage alone:

$$prior_{cit} = \frac{\prod_{j=1}^{J} post_{cjt} * r_{ij} + (1 - r_{ij})/(K+1)}{\sum_{c=0}^{K} \prod_{j=1}^{J} post_{cjt} * r_{ij} + (1 - r_{ij})/(K+1)}$$

For mapping populations, `AddGenotypePriorProb_LD` uses the above formula when each allele only has two possible genotypes (i.e. test-cross segregation). When more genotypes are possible, `AddGenotypePriorProb_LD` instead estimates prior probabilities as fitted values when the posterior

probabilities for a given allele are regressed on the posterior probabilities for a linked allele. This allows loci with different segregation patterns to be informative for predicting genotypes, and for cases where two alleles are in phase for some but not all parental copies.

Value

A RADdata object is returned. For AddAlleleLinkages, it has a new slot called $alleleLinkages that is a list, with one item in the list for each allele in the dataset. Each item is a data frame, with indices for linked alleles in the first column, and correlation coefficients in the second column.

For AddGenotypePriorProb_LD, the object has a new slot called $priorProbLD. This is a list much like $posteriorProb, with one list item per inheritance mode, and each item being an array with allele copy number in the first dimension, taxa in the second dimension, and alleles in the third dimension. Values indicate genotype prior probabilities based on linked alleles alone.

Author(s)

Lindsay V. Clark

See Also

AddGenotypePriorProb_HWE

Examples

```
# load example dataset
data(Msi01genes)

# Run non-LD pop structure pipeline
Msi01genes <- IteratePopStruct(Msi01genes, tol = 0.01, nPcsInit = 10)

# Add linkages
Msi01genes <- AddAlleleLinkages(Msi01genes, "popstruct", 1e4, 0.05)
# Get new posterior probabilities based on those linkages
Msi01genes <- AddGenotypePriorProb_LD(Msi01genes, "popstruct")

# Preview results
Msi01genes$priorProbLD[[1]][,1:10,1:10]
```

AddGenotypeLikelihood *Estimate Genotype Likelihoods in a RADdata object*

Description

For each possible allele copy number across each possible ploidy in each taxon, AddGenotypeLikelihood estimates the probability of observing the distribution of read counts that are recorded for that taxon and locus.

Usage

```
AddGenotypeLikelihood(object, ...)

## S3 method for class 'RADdata'
AddGenotypeLikelihood(object, overdispersion = 9, ...)
```

Arguments

`object`	A "RADdata" object.
`overdispersion`	An overdispersion parameter. Higher values will cause the expected read depth distribution to more resemble the binomial distribution. Lower values indicate more overdispersion, *i.e.* sample-to-sample variance in the probability of observing reads from a given allele.
`...`	Other arguments; none are currently used.

Details

If allele frequencies are not already recorded in `object`, they will be added using `AddAlleleFreqHWE`. Allele frequencies are then used for estimating the probability of sampling an allele from a genotype due to sample contamination. Given a known genotype with x copies of allele i, ploidy k, allele frequency p_i in the population used for making sequencing libraries, and contamination rate c, the probabiity of sampling a read r_i from that locus corresponding to that allele is

$$P(r_i|x) = \frac{x}{k} * (1 - c) + p_i * c$$

To estimate the genotype likelihood, where nr_i is the number of reads corresponding to allele i for a given taxon and locus and nr_j is the number of reads corresponding to all other alleles for that taxon and locus:

$$P(nr_i, nr_j|x) = \binom{nr_i + nr_j}{nr_i} * \frac{B[P(r_i|x) * d + nr_i, [1 - P(r_i|x)] * d + nr_j]]}{B[P(r_i|x) * d, [1 - P(r_i|x)] * d]}$$

where

$$\binom{nr_i + nr_j}{nr_i} = \frac{(nr_i + nr_j)!}{nr_i! * nr_j!}$$

B is the beta function, and d is the overdispersion parameter set by `overdispersion`.

Value

A "RADdata" object identical to that passed to the function, but with genotype likelihoods stored in `object$genotypeLikelihood`. This item is a list, with one item for each possible ploidy, ignoring differences between autopolyploids and allopolyploids. For each ploidy there is a three-dimensional array with number of allele copies in the first dimension, taxa in the second dimension, and alleles in the third dimension.

Author(s)

Lindsay V. Clark

See Also

AddAlleleFreqMapping

Examples

```
# load example dataset and add allele frequency
data(exampleRAD)
exampleRAD <- AddAlleleFreqHWE(exampleRAD)

# estimate genotype likelihoods
exampleRAD <- AddGenotypeLikelihood(exampleRAD)

# inspect the results
# the first ten individuals and first two alleles, assuming diploidy
exampleRAD$alleleDepth[1:10,1:2]
exampleRAD$genotypeLikelihood[[1]][,1:10,1:2]
# assuming tetraploidy
exampleRAD$genotypeLikelihood[[2]][,1:10,1:2]
```

AddGenotypePosteriorProb

Estimate Posterior Probabilities of Genotypes

Description

Given a "RADdata" object containing genotype prior probabilities and genotype likelihoods, this function estimates genotype posterior probabilities and adds them to the $posteriorProb slot of the object.

Usage

```
AddGenotypePosteriorProb(object, ...)
```

Arguments

object	A "RADdata" object. Prior genotype probabilities and genotype likelihood should have already been added.
...	Potential future arguments (none currently in use).

Details

If AddPriorTimesLikelihood has not already been run on the object, it will be run by AddGenotypePosteriorProb in order to perform the necessary calculations.

Value

A "RADdata" object identical to that passed to the function, but with a list added to the $posteriorProb slot. Each item of the list is a three dimensional array, with allele copy number in the first dimension, taxa in the second dimension, and alleles in the third dimension. For each allele and taxa, posterior probabilities will sum to one across all potential allele copy numbers. There will be one such array for each possible ploidy, corresponding to object$priorProb.

Author(s)

Lindsay V. Clark

See Also

AddGenotypeLikelihood, AddGenotypePriorProb_Mapping2Parents

Examples

```
# load dataset and set some parameters
data(exampleRAD_mapping)
exampleRAD_mapping <- SetDonorParent(exampleRAD_mapping, "parent1")
exampleRAD_mapping <- SetRecurrentParent(exampleRAD_mapping, "parent2")
exampleRAD_mapping <- AddAlleleFreqMapping(exampleRAD_mapping,
                                           expectedFreqs = c(0.25, 0.75),
                                           allowedDeviation = 0.08)
exampleRAD_mapping <- AddGenotypeLikelihood(exampleRAD_mapping)
exampleRAD_mapping <- AddGenotypePriorProb_Mapping2Parents(exampleRAD_mapping,
                                                           n.gen.backcrossing = 1)

# estimate posterior probabilities
exampleRAD_mapping <- AddGenotypePosteriorProb(exampleRAD_mapping)
# examine the results
exampleRAD_mapping$posteriorProb[[1]][,3,]
```

AddGenotypePriorProb_ByTaxa

Estimate Prior Genotype Probabilities on a Per-Taxon Basis

Description

Using local allele frequencies estimated by AddAlleleFreqByTaxa and assuming Hardy-Weinberg Equilibruim or inbreeding on a local scale, AddGenotypePriorProb_ByTaxa estimates prior genotype probabilities at each taxon, allele, and possible ploidy. These are then stored in the $priorProb slot of the "RADdata" object.

Usage

```
AddGenotypePriorProb_ByTaxa(object, ...)
## S3 method for class 'RADdata'
AddGenotypePriorProb_ByTaxa(object, selfing.rate = 0, ...)
```

Arguments

object	A "RADdata" object. AddAlleleFreqByTaxa should have already been run.
selfing.rate	A number ranging from zero to one indicating the frequency of self-fertilization in the species.
...	Additional arguments (none implemented).

Value

A "RADdata" object identical to that passed to the function, but with a list added to the $priorProb slot. Each item in the list corresponds to one ploidy in object$possiblePloidies, and is a three-dimensional array with allele copy number in the first dimension, taxa in the second dimension, and alleles in the third dimension. The values in the array are prior genotype probabilities. Additionally, object$possiblePloidies is copied to object$priorProbPloidies, and "taxon" is recorded in the "priorType" attribute.

Author(s)

Lindsay V. Clark

See Also

AddGenotypePriorProb_HWE for equations used for genotype prior probability estimation.

AddGenotypePriorProb_Mapping2Parents, AddGenotypeLikelihood

Examples

```
# load data
data(exampleRAD)
# do PCA
exampleRAD <- AddPCA(exampleRAD, nPcsInit = 3)
# get allele frequencies
exampleRAD <- AddAlleleFreqByTaxa(exampleRAD)

# add prior probabilities
exampleRAD <- AddGenotypePriorProb_ByTaxa(exampleRAD)

exampleRAD$priorProb[[1]][,1,]
exampleRAD$priorProb[[2]][,1,]
exampleRAD$priorProb[[1]][,2,]
exampleRAD$priorProb[[2]][,2,]

# try it with inbreeding
exampleRAD <- AddGenotypePriorProb_ByTaxa(exampleRAD, selfing.rate = 0.5)

exampleRAD$priorProb[[1]][,1,]
```

AddGenotypePriorProb_Even

Add Uniform Priors to a RADdata Object

Description

To estimate genotype posterior probabilities based on read depth alone, without taking any population parameters into account, this function can be used to set a uniform prior probability on all possible genotypes. This function is not part of any pipeline but can be used for very rough and quick genotype estimates, when followed by AddGenotypeLikelihood, AddGenotypePosteriorProb, AddPloidyChiSq, and GetWeightedMeanGenotypes or GetProbableGenotypes.

Usage

```
AddGenotypePriorProb_Even(object, ...)
```

Arguments

object	A RADdata object.
...	Additional arguments (none implemented).

Value

A "RADdata" object identical that passed to the function, but with data stored in two new slots:

priorProb	A list of matrices, with one matrix per possible ploidy. For each matrix, allele copy number (from zero to the total ploidy) is in rows, and alleles are in columns. Each value is $1/(ploidy + 1)$.
priorProbPloidies	A list identical to object$possiblePloidies. It is in the same order as $priorProb, with each item indicating the inheritance mode for the corresponding prior probability matrix.

Note

Values in object$ploidyChiSq may not be particularly meaningful under uniform priors.

Author(s)

Lindsay V. Clark

See Also

AddGenotypePriorProb_HWE

Examples

```
data(exampleRAD)

exampleRAD <- AddGenotypePriorProb_Even(exampleRAD)
exampleRAD$priorProb

# finish protocol to get genotypes
exampleRAD <- AddGenotypeLikelihood(exampleRAD)
exampleRAD <- AddPloidyChiSq(exampleRAD)
exampleRAD <- AddGenotypePosteriorProb(exampleRAD)

genmat <- GetWeightedMeanGenotypes(exampleRAD)
genmat
```

AddGenotypePriorProb_HWE

Estimate Genotype Prior Probabilities In the Absence of Population Structure

---

Description

Assuming Hardy-Weinberg Equilibrium, this function uses allele frequencies and possible ploidies stored in a "RADdata" object to estimate genotype frequencies in the population, then stores these genotype frequencies in the `$priorProb` slot. Inbreeding can also be simulated using the `selfing.rate` argument.

Usage

```
AddGenotypePriorProb_HWE(object, ...)
## S3 method for class 'RADdata'
AddGenotypePriorProb_HWE(object, selfing.rate = 0, ...)
```

Arguments

`object`	A "RADdata" object that has had allele frequencies added with `AddAlleleFreqHWE`.
`selfing.rate`	A number ranging from zero to one indicating the frequency of self-fertilization in the species.
`...`	Additional arguments (none currently implemented).

Details

For an autopolyploid, or within one subgenome of an allopolyploid, genotype prior probabilities are estimated as:

$$P(G_i) = \binom{k}{i} p^i * (1-p)^{k-i}$$

where k is the ploidy, i is the copy number of a given allele, and p is the allele frequency in the population.

If the selfing rate is above zero, genotype prior probabilities are asjusted according to Equation 6 of de Silva et al. (2005):

$$P(G_{self}) = (1 - s)(I - sA)^{-1}P(G)$$

where s is the selfing rate. A is a $k + 1 \times k + 1$ matrix, with each column representing the allele copy number from 0 to k of a parental genotype, and each row representing the allele copy number from 0 to k of a progeny genotype, and matrix elements representing the frequencies of progeny after self-fertilization (each column summing to one).

Value

A "RADdata" object identical that passed to the function, but with data stored in two new slots:

`priorProb` A list of matrices, with one matrix per possible ploidy. For each matrix, allele copy number (from zero to the total ploidy) is in rows, and alleles are in columns. Each value is the probability of sampling an individual with that allele copy number from the population.

`priorProbPloidies` A list identical to `object$possiblePloidies`. It is in the same order as `$priorProb`, with each item indicating the inheritance mode for the corresponding prior probability matrix.

Author(s)

Lindsay V. Clark

References

De Silva, H. N., Hall, A. J., Rikkerink, E., and Fraser, L. G. (2005) Estimation of allele frequencies in polyploids under certain patterns of inheritance. *Heredity* **95**, 327–334. doi:10.1038/sj.hdy.6800728

See Also

`AddGenotypePriorProb_Mapping2Parents`, `AddGenotypeLikelihood`, `AddGenotypePriorProb_ByTaxa`

Examples

```
# load in an example dataset
data(exampleRAD)
# add allele frequencies
exampleRAD <- AddAlleleFreqHWE(exampleRAD)
# add inheritance modes
exampleRAD$possiblePloidies <- list(2L, 4L, c(2L, 2L))

# estimate genotype prior probabilities
exampleRAD <- AddGenotypePriorProb_HWE(exampleRAD)
```

```
# examine results
exampleRAD$alleleFreq
exampleRAD$priorProb

# try it with inbreeding
exampleRAD2 <- AddGenotypePriorProb_HWE(exampleRAD, selfing.rate = 0.5)
exampleRAD2$priorProb
```

AddGenotypePriorProb_Mapping2Parents

Expected Genotype Frequencies in Mapping Populations

Description

Given the most likely genotypes of two parent taxa, expected genotype frequencies are estimated for a population of progeny and are added to the "RADdata" object in the $priorProb slot.

Usage

```
AddGenotypePriorProb_Mapping2Parents(object, ...)
## S3 method for class 'RADdata'
AddGenotypePriorProb_Mapping2Parents(object,
    donorParent = GetDonorParent(object),
    recurrentParent = GetRecurrentParent(object),
    n.gen.backcrossing = 0, n.gen.intermating = 0, n.gen.selfing = 0,
    donorParentPloidies = object$possiblePloidies,
    recurrentParentPloidies = object$possiblePloidies,
    minLikelihoodRatio = 10, ...)
```

Arguments

object	A "RADdata" object. Ideally this should be set up as a mapping population using SetDonorParent, SetRecurrentParent, and AddAlleleFreqMapping.
...	Additional arguments, listed below, to be passed to the method for "RADdata" objects.
donorParent	A character string indicating which taxon is the donor parent. If backcrossing was not performed, it does not matter which was the donor or recurrent parent.
recurrentParent	A character string indicating which taxon is the recurrent parent.
n.gen.backcrossing	An integer, zero or greater, indicating how many generations of backcrossing to the recurrent parent were performed.
n.gen.intermating	An integer, zero or greater, indicating how many generations of intermating within the population were performed. (Values above one should not have an effect on the genotype priors that are output, *i.e.* genotype probabilities after one generation of random mating are identical to genotype probabilities after >1 generation of random mating, assuming no genetic drift or selection).

`n.gen.selfing` An integer, zero or greater, indicating how many generations of selfing were performed.

`donorParentPloidies`
A list, where each item in the list is an integer vector indicating a potential inheritance mode that could be observed among loci in the donor parent. `2` indicates diploid, `4` indicates autotetraploid, `c(2, 2)` indicates allotetraploid, *etc.*

`recurrentParentPloidies`
A list in the same format as `donorParentPloidies` indicating inheritance modes that could be observed among loci in the recurrent parent.

`minLikelihoodRatio`
The minimum likelihood ratio for determining parental genotypes with confidence, to be passed to `GetLikelyGen` for both parental taxa.

Details

The function first determines which combinations of inheritance modes from the two parents should be examined in the progeny. The expected progeny ploidy must be in `object$possiblePloidies` for a given combination to be examined.

The most likely genotypes for the two parents are estimated using `GetLikelyGen`. If parental gentoypes don't match progeny allele frequencies, the function attempts to correct the parental genotypes to the most likely combination that matches the allele frequency. For each ploidy being examined, F1 genotype probabilities are then calculated.

Genotype probabilities are updated for each backcrossing generation, then each intermating generation, then each selfing generation.

The default, with `n.gen.backcrossing = 0`, `n.gen.intermating = 0` and `n.gen.selfing = 0`, will simulate an F1 population. A BC1F2 population, for example, would have `n.gen.backcrossing = 1`, `n.gen.intermating = 0` and `n.gen.selfing = 1`. A typical F2 population would have `n.gen.selfing = 1` and the other two parameters set to zero. However, in a self-incompatible species where many F1 are intermated to produce the F2, one would instead use `n.gen.intermating = 1` and set the other parameters to zero.

Value

A `"RADdata"` object identical to that passed to the function, but with data stored in six new slots:

`priorProb` A list of matrices, with one matrix per possible ploidy of offspring. For each matrix, allele copy number (from zero to the total ploidy) is in rows, and alleles are in columns. Each value is the probability of sampling an individual with that allele copy number from the population.

`priorProbPloidies`
A list in the same format as `object$possiblePloidies`, and the same length as `object$priorProb`. Each item in the list is a vector indicating the inheritance mode for the corresponding matrix in `object$priorProb`.

`donorPloidies` A list in the same format as `object$possiblePloidies`, with one item corresponding to each in object$priorProbPloidies, indicating the donor parent ploidy for that progeny ploidy.

`recurrentPloidies`
: A list in the same format as `object$possiblePloidies`, with one item corresponding to each in `object$priorProbPloidies`, indicating the recurrent parent ploidy for that progeny ploidy.

`likelyGeno_donor`
: A matrix of the donor parent genotypes that were used for estimating genotype prior probabilities. Formatted like the output of `GetLikelyGen`.

`likelyGeno_recurrent`
: A matrix of the recurrent parent genotypes that were use for estimating gentoype prior probabilities.

Note

For the time being, in allopolyploids it is assumed that copies of an allele are distributed among as few isoloci as possible. For example, if an autotetraploid genotype had two copies of allele A and two copies of allele B, it is assumed to be AA BB rather than AB AB. This may be remedied in the future by examining distribution of genotype likelihoods.

Author(s)

Lindsay V. Clark

See Also

`AddGenotypeLikelihood`, `AddGenotypePriorProb_HWE`

Examples

```
# load dataset and set some parameters
data(exampleRAD_mapping)
exampleRAD_mapping <- SetDonorParent(exampleRAD_mapping, "parent1")
exampleRAD_mapping <- SetRecurrentParent(exampleRAD_mapping, "parent2")
exampleRAD_mapping <- AddAlleleFreqMapping(exampleRAD_mapping,
                                           expectedFreqs = c(0.25, 0.75),
                                           allowedDeviation = 0.08)
exampleRAD_mapping <- AddGenotypeLikelihood(exampleRAD_mapping)

# examine the dataset
exampleRAD_mapping
exampleRAD_mapping$alleleFreq

# estimate genotype priors for a BC1 population
exampleRAD_mapping <- AddGenotypePriorProb_Mapping2Parents(exampleRAD_mapping,
                                                           n.gen.backcrossing = 1)
exampleRAD_mapping$priorProb
```

AddPCA *Perform Principal Components Analysis on "RADdata" Object*

Description

This function uses read depth ratios or posterior genotype probabilities (the latter preferentially) as input data for principal components analysis. The PCA scores are then stored in the `$PCA` slot of the "RADdata" object.

Usage

```
AddPCA(object, ...)
## S3 method for class 'RADdata'
AddPCA(object, nPcsInit = 10, maxR2changeratio = 0.05,
                          minPcsOut = 1, ...)
```

Arguments

`object`	A "RADdata" object.
`nPcsInit`	The number of principal component axes to initially calculate.
`maxR2changeratio`	This number determines how many principal component axes are retained. The difference in R^2 values between the first and second axes is multiplied by `maxR2changeratio`. The last axis retained is the first axis after which the R^2 value changes by less than this value. Lower values of `maxR2changeratio` will result in more axes being retained.
`minPcsOut`	The minimum number of PC axes to output, which can override `maxR2changeratio`.
`...`	Additional arguments to be passed to the `pca` function from the **pcaMethods** BioConductor package.

Details

The PPCA (probabalistic PCA) method from **pcaMethods** is used, due to the high missing data rate that is typical of genotyping-by-sequencing datasets.

Value

A "RADdata" object identical to the one passed to the function, but with a matrix added to the `$PCA` slot. This matrix contains PCA scores, with taxa in rows, and PC axes in columns.

Note

If you see the error

```
Error in if (rel_ch < threshold & count > 5) { :   missing value where TRUE/FALSE needed
```

try lowering `nPcsInit`.

Author(s)

Lindsay V. Clark

See Also

AddAlleleFreqByTaxa

Examples

```
# load data
data(exampleRAD)
# do PCA
exampleRAD <- AddPCA(exampleRAD, nPcsInit = 3)

plot(exampleRAD$PCA[,1], exampleRAD$PCA[,2])
```

AddPloidyChiSq *Chi-Square Test on Genotype Likelihood Distributions*

Description

This function is intended to help identify the correct inheritance mode for each locus in a "RADdata" object. Expected genotype frequencies are taken from `object$priorProb`. Observed genotype frequencies are estimated from `object$genotypeLikelihood`, where each taxon has a partial assignment to each genotype, proportional to genotype likelihoods. A χ^2 statistic is then estimated.

Usage

```
AddPloidyChiSq(object, ...)
## S3 method for class 'RADdata'
AddPloidyChiSq(object, excludeTaxa = GetBlankTaxa(object),
               ...)
```

Arguments

`object`	A "RADdata" object. Genotype prior probabilities and likelihoods should have been added.
`excludeTaxa`	A character vector indicating names of taxa to exclude from calculations.
`...`	Additional arguments to be passed to other methods (none currently in use).

Details

Parents (in mapping populations) and blank taxa are automatically excluded from calculations.

Genotypes with zero prior probability would result in an infinite A χ^2 statistic and therefore are excluded from the calculation. However, the total number of observations (total number of taxa) remains the same, so that if there are many taxa with high likelihood for a genotype with zero prior probability, χ^2 will be high.

Value

A "RADdata" object identical to the one passed to the function, but with matrices added to the $ploidyChiSq and $ploidyChiSqP slots. Both matrices have ploidies (matching object$priorProb) in rows and alleles in columns. object$ploidyChiSq contains the χ^2 values. object$ploidyChiSqP contains p-values, *i.e.* the probability of genotype distributions deviating that far from expectations if the expectations are correct. These p-values may be relatively low due to genotype uncertainty.

Author(s)

Lindsay V. Clark

See Also

AddGenotypeLikelihood, AddPloidyLikelihood

Examples

```
# load dataset and set some parameters
data(exampleRAD_mapping)
exampleRAD_mapping <- SetDonorParent(exampleRAD_mapping, "parent1")
exampleRAD_mapping <- SetRecurrentParent(exampleRAD_mapping, "parent2")
exampleRAD_mapping <- AddAlleleFreqMapping(exampleRAD_mapping,
                                           expectedFreqs = c(0.25, 0.75),
                                           allowedDeviation = 0.08)
exampleRAD_mapping <- AddGenotypeLikelihood(exampleRAD_mapping)
exampleRAD_mapping <- AddGenotypePriorProb_Mapping2Parents(exampleRAD_mapping,
                                                           n.gen.backcrossing = 1)

# get chi-squared values
exampleRAD_mapping <- AddPloidyChiSq(exampleRAD_mapping)
# view chi-squared and p-values (diploid only)
exampleRAD_mapping$ploidyChiSq
exampleRAD_mapping$ploidyChiSqP
```

AddPloidyLikelihood *Likelihoods for Possible Ploidies Based on Genotype Distributions*

Description

Given prior genotype probabilities, and a set of high-confidence genotypes estimated with GetLikelyGen, this function estimates the probability of observing that distribution of genotypes and stores the probability in the $ploidyLikelihood slot of the "RADdata" object.

Usage

```
AddPloidyLikelihood(object, ...)
## S3 method for class 'RADdata'
AddPloidyLikelihood(object, excludeTaxa = GetBlankTaxa(object),
                            minLikelihoodRatio = 50, ...)
```

Arguments

object	A "RADdata" object. Prior genotype probabilities and genotype likelihoods should have already been added using the appropriate functions.
...	Additional arguments to be passed to the method for "RADdata".
excludeTaxa	A character vector indicating taxa that should be excluded from calculations.
minLikelihoodRatio	A number, one or higher, to be passed to GetLikelyGen.

Details

The purpose of this function is to estimate the correct inheritance mode for each locus. This function may be deleted in the future in favor of better alternatives.

Value

A "RADdata" object identical to that passed to the function, but with results added to the $ploidyLikelihood slot. This has one row for each possible ploidy (each ploidy with data in $priorProb), and one column for each allele. Each element of the matrix is the multinomial probability of seeing that distribution of genotypes given the prior probabilities.

Author(s)

Lindsay V. Clark

See Also

AddPloidyChiSq

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (object, ...)
{
    UseMethod("AddPloidyLikelihood", object)
  }
```

AddPriorTimesLikelihood *Multiply Genotype Prior Probabilities by Genotype Likelihoods*

Description

In preparation for calculations to be done by other functions, the genotype likelihoods for each taxon, allele, and ploidy are multiplied by the corresponding genotype prior probabilities, then stored back in the "RADdata" object in the $priorTimesLikelihood slot.

Usage

```
AddPriorTimesLikelihood(object, ...)
```

Arguments

object	A "RADdata" object. Prior genotype probabilities and genotype likelihood should have already been added.
...	Potential future arguments (none currently in use).

Value

A "RADdata" object identical to that passed to the function, but with the $priorTimesLikelihood slot added. This slot contains a list. Each element in the list corresponds to an element in object$priorProbs. Each element is an array, with allele copy number in the first dimension, taxa in the second dimension, and alleles in the third dimension.

Author(s)

Lindsay V. Clark

See Also

AddGenotypeLikelihood, AddGenotypePriorProb_Mapping2Parents

Examples

```
# load dataset and set some parameters
data(exampleRAD_mapping)
exampleRAD_mapping <- SetDonorParent(exampleRAD_mapping, "parent1")
exampleRAD_mapping <- SetRecurrentParent(exampleRAD_mapping, "parent2")
exampleRAD_mapping <- AddAlleleFreqMapping(exampleRAD_mapping,
                                           expectedFreqs = c(0.25, 0.75),
                                           allowedDeviation = 0.08)
exampleRAD_mapping <- AddGenotypeLikelihood(exampleRAD_mapping)
exampleRAD_mapping <- AddGenotypePriorProb_Mapping2Parents(exampleRAD_mapping,
                                                           n.gen.backcrossing = 1)
```

```
# perform the multiplication
exampleRAD_mapping <- AddPriorTimesLikelihood(exampleRAD_mapping)
# examine the results
exampleRAD_mapping$priorTimesLikelihood[[1]][,50,] # for one progeny
exampleRAD_mapping$priorTimesLikelihood[[1]][,1,]
 # --> for the donor parent; not a good idea to use since the priors
 #    aren't appropriate
```

CanDoGetWeightedMeanGeno *Check Whether* GetWeightedMeanGenotypes *Can Be Run*

Description

This function is used internally by AddPCA, AddAlleleFreqByTaxa, and the internal function .alleleFreq to test whether GetWeightedMeanGenotypes can be run on a "RADdata" object.

Usage

```
CanDoGetWeightedMeanGeno(object, ...)
```

Arguments

object	A "RADdata" object.
...	Additional arguments (none implemented).

Value

A single Boolean value. To be TRUE, object$posteriorProb must be non-null, and either there must be only one possible ploidy, or object$ploidyChiSq must be non-null.

Author(s)

Lindsay V. Clark

See Also

AddGenotypePosteriorProb, AddPloidyChiSq

Examples

```
data(exampleRAD)

CanDoGetWeightedMeanGeno(exampleRAD)

exampleRAD <- AddAlleleFreqHWE(exampleRAD)
exampleRAD <- AddGenotypePriorProb_HWE(exampleRAD)
exampleRAD <- AddGenotypeLikelihood(exampleRAD)
exampleRAD <- AddPloidyChiSq(exampleRAD)
```

```
exampleRAD <- AddGenotypePosteriorProb(exampleRAD)

CanDoGetWeightedMeanGeno(exampleRAD)
```

`EstimateContaminationRate` *Estimate Sample Contamination Using Blanks*

Description

Based on mean read depth at blank and non-blank taxa, estimate sample cross-contamination and add that information to the "RADdata" object.

Usage

```
EstimateContaminationRate(object, ...)
## S3 method for class 'RADdata'
EstimateContaminationRate(object, multiplier = 1, ...)
```

Arguments

`object`	A "RADdata" object where SetBlankTaxa has already been used to assign one or more taxa as blanks.
`multiplier`	A single numeric value, or a named numeric vector with one value per blank taxon in `object`, with names matching the blank taxa names. Read depth at blank taxa will be multiplied by this number when estimating sample cross-contamination. See example below.
`...`	Additional arguments (none implemented).

Details

This function estimates sample cross-contamination assuming that the only source of contamination is from adapter or sample spill-over between wells during library preparation, or contamination among the libraries themselves. If you anticipate a higher rate of contamination during DNA extraction before library preparation, you may wish to increase the value using SetContamRate.

It is important to set the contamination rate to a reasonably accurate value (*i.e.* the right order of magnitude) in order for **polyRAD** to be able to identify homozygotes that may otherwise appear heterozygous due to contamination.

Value

A "RADdata" object identical to `object` but with the "contamRate" attribute adjusted.

Author(s)

Lindsay V. Clark

Examples

```
# dataset for this example
data(Msi01genes)

# give the name of the taxon that is blank
Msi01genes <- SetBlankTaxa(Msi01genes, "blank")

# Fifteen libraries were done; blank is pooled over all of them, and
# most other samples are pooled over two libraries.
mymult <- 2/15

# estimate the contamination rate
Msi01genes <- EstimateContaminationRate(Msi01genes, multiplier = mymult)
```

---

exampleRAD *Miniature Datasets for Testing polyRAD Functions*

Description

exampleRAD and exampleRAD_mapping are two very small simulated "RADdata" datasets for testing polyRAD functions. Each has four loci. exampleRAD is a natural population of 100 individuals with a mix of diploid and tetraploid loci. exampleRAD_mapping is a diploid BC1 mapping population with two parents and 100 progeny. Msi01genes is a "RADdata" object with 585 taxa and 24 loci, containing real data from *Miscanthus sinensis*, obtained by using VCF2RADdata on the file Msi01genes.vcf.

Usage

```
data(exampleRAD)
data(exampleRAD_mapping)
data(Msi01genes)
```

Format

See the format described in "RADdata".

Source

Randomly generated using a script available in polyRAD/extdata/simulate_rad_data.R.

M. sinensis sequencing data available at https://www.ncbi.nlm.nih.gov//bioproject/PRJNA207721.

Examples

```
data(exampleRAD)
exampleRAD
data(exampleRAD_mapping)
exampleRAD_mapping
data(Msi01genes)
Msi01genes
```

ExportGAPIT *Export RADdata Object for Use by Other R Packages*

Description

After a "RADdata" object has been run through a pipeline such as IteratePopStruct, these functions can be used to export the genotypes to R packages and other software that can perform genome-wide association and genomic prediction. ExportGAPIT, Export_rrBLUP_Amat, Export_rrBLUP_GWAS, and Export_TASSEL_Numeric all export continuous numerical genotypes generated by GetWeightedMeanGenotypes. Export_polymapR and Export_GWASpoly use GetProbableGenotypes to export discrete genotypes. Export_MAPpoly writes genotype posterior probabilities to a file for import by **MAPpoly**.

Usage

```
ExportGAPIT(object, onePloidyPerAllele = FALSE)
Export_rrBLUP_Amat(object, naIfZeroReads = FALSE,
                   onePloidyPerAllele = FALSE)
Export_rrBLUP_GWAS(object, naIfZeroReads = FALSE,
                   onePloidyPerAllele = FALSE)
Export_TASSEL_Numeric(object, file, naIfZeroReads = FALSE,
                      onePloidyPerAllele = FALSE)
Export_polymapR(object, naIfZeroReads = TRUE,
                progeny = GetTaxa(object)[!GetTaxa(object) %in%
                  c(GetDonorParent(object), GetRecurrentParent(object),
                    GetBlankTaxa(object))])
Export_MAPpoly(object, file, pheno = NULL, ploidyIndex = 1,
               progeny = GetTaxa(object)[!GetTaxa(object) %in%
                 c(GetDonorParent(object), GetRecurrentParent(object),
                   GetBlankTaxa(object))],
               digits = 3)
Export_GWASpoly(object, file, naIfZeroReads = TRUE)
```

Arguments

object	A "RADdata" object with posterior genotype probabilities already estimated.
onePloidyPerAllele	Logical. If TRUE, for each allele the inheritance mode with the lowest χ^2 value is selected and is assumed to be the true inheritance mode. If FALSE, inheritance modes are weighted by inverse χ^2 values for each allele, and mean genotypes that have been weighted across inheritance modes are returned.
naIfZeroReads	A logical indicating whether NA should be inserted into the output matrix for any taxa and loci where the total read depth for the locus is zero. If FALSE, the output for these genotypes is essentially the mode (for Export_polymapR and Export_GWASpoly) or mean (for others) across prior genotype probabilities, since prior and posterior genotype probabilities are equal when there are no reads.

file	A character string indicating a file path to which to write.
pheno	A data frame or matrix of phenotypic values, with progeny in rows and traits in columns. Columns should be named.
ploidyIndex	Index, within object$priorProbPloidies, of the ploidy to be exported.
progeny	A character vector indicating which individuals to export as progeny of the cross.
digits	Number of decimal places to which to round genotype probabilities in the output file.

Details

GAPIT, **FarmCPU**, **rrBLUP**, and **TASSEL** allow genotypes to be a continuous numeric variable. **MAPpoly** allows for import of genotype probabilities. **GAPIT** does not allow missing data, hence there is no naIfZeroReads argument for ExportGAPIT. Genotypes are exported on a scale of -1 to 1 for **rrBLUP**, on a scale of 0 to 2 for **GAPIT** and **FarmCPU**, and on a scale of 0 to 1 for **TASSEL**.

For all functions, one allele per marker is dropped. Export_MAPpoly also drops alleles where one or both parental genotypes could not be determined, and where both parents are homozygotes.

For ExportGAPIT and Export_rrBLUP_GWAS, chromosome and position are filled with dummy data if they do not exist in object$locTable. For Export_TASSEL_Numeric, allele names are exported, but no chromosome or position information per se.

If the chromosomes in object$locTable are in character format, ExportGAPIT and Export_MAPpoly will attempt to extract chromosome numbers.

Because **polymapR** allows only one ploidy, there must only be one possible ploidy for the progeny in the RADdata object (possibly with different ploidies for the parents, *e.g.* 4x x 2x = 3x). **MAPpoly** also only allows one ploidy, but Export_MAPpoly allows the user to select which ploidy from the RADdata object to use. (This is due to how internal **polyRAD** functions are coded.)

Value

For ExportGAPIT, a list:

GD	A data frame with taxa in the first column and alleles (markers) in subsequent columns, containing the genotypes. To be passed to the GD argument for GAPIT or FarmCPU.
GM	A data frame with the name, chromosome number, and position of every allele (marker). To be passed to the GM argument for GAPIT or FarmCPU.

For Export_rrBLUP_Amat, a matrix with taxa in rows and alleles (markers) in columns, containing genotype data. This can be passed to A.mat in **rrBLUP**.

For Export_rrBLUP_GWAS, a data frame with alleles (markers) in rows. The first three columns contain the marker names, chromosomes, and positions, and the remaining columns each represent one taxon and contain the genotype data. This can be passed to the GWAS function in **rrBLUP**.

Export_TASSEL_Numeric and Export_MAPpoly write a file but does not return an object.

For Export_polymapR, a matrix of integers indicating the most probable allele copy number, with markers in rows and individuals in columns. The parents are listed first, followed by all progeny.

Export_MAPpoly and Export_GWASpoly write files but do not return an object. Files output by Export_GWASpoly are comma delimited and in numeric format.

Note

rrBLUP and **polymapR** are available through CRAN, and **GAPIT** and **FarmCPU** must be downloaded from the Zhang lab website. **MAPpoly** is available on GitHub but not yet on CRAN. **GWASpoly** is available from the Endelman lab website.

In my experience with **TASSEL 5**, numerical genotype files that are too large do not load/display properly. If you run into this problem I recommend using `SplitByChromosome` to split your `RADdata` object into multiple smaller objects, which can then be exported to separate files using `Export_TASSEL_Numeric`. If performing GWAS, you may also need to compute a kinship matrix using separate software such as **rrBLUP**.

Author(s)

Lindsay V. Clark

References

http://zzlab.net/GAPIT/

Lipka, A. E., Tian, F., Wang, Q., Peiffer, J., Li, M., Bradbury, P. J., Gore, M. A., Buckler, E. S. and Zhang, Z. (2012) GAPIT: genome association and prediction integrated tool. *Bioinformatics* **28**, 2397–2399.

http://www.zzlab.net/FarmCPU/

Liu, X., Huang, M., Fan, B., Buckler, E. S., Zhang, Z. (2016) Iterative usage of fixed and random effects models for powerful and efficient genome-wide association studies. *PLoS Genetics* **12**, e1005767.

Endelman, J.B. (2011) Ridge Regression and Other Kernels for Genomic Selection with R Package rrBLUP. *The Plant Genome* **4**, 250–255.

http://www.maizegenetics.net/tassel

Bradbury, P. J., Zhang, Z., Kroon, D. E., Casstevens, T. M., Ramdoss, Y. and Buckler, E. S. (2007) TASSEL: Software for association mapping of complex traits in diverse samples. *Bioinformatics* **23**, 2633–2635.

Bourke, P., van Geest, G., Voorrips, R. E., Jansen, J., Kranenberg, T., Shahin, A., Visser, R. G. F., Arens, P., Smulders, M. J. M. and Maliepaard, C. (2018) polymapR: linkage analysis and genetic map construction from F1 populations of outcrossing polyploids. *Bioinformatics* **34**, 3496–3502.

https://github.com/mmollina/MAPpoly

Mollinari, M. and Garcia, A. A. F. (2018) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models. *bioRxiv* doi: https://doi.org/10.1101/415232.

https://potatobreeding.cals.wisc.edu/software/

Rosyara, U. R., De Jong, W. S., Douches, D. S., and Endelman, J. B. (2016) Software for Genome-Wide Association Studies in Autopolyploids and Its Application to Potato. *Plant Genome* **9**.

See Also

`GetWeightedMeanGenotypes`

Examples

```
# load example dataset
data(exampleRAD)
# get genotype posterior probabilities
exampleRAD <- IterateHWE(exampleRAD)

# export to GAPIT
exampleGAPIT <- ExportGAPIT(exampleRAD)

# export to rrBLUP
example_rrBLUP_A <- Export_rrBLUP_Amat(exampleRAD)
example_rrBLUP_GWAS <- Export_rrBLUP_GWAS(exampleRAD)

# export to TASSEL
outfile <- tempfile() # temporary file for example
Export_TASSEL_Numeric(exampleRAD, outfile)

# for mapping populations
data(exampleRAD_mapping)

# specify donor and recurrent parents
exampleRAD_mapping <- SetDonorParent(exampleRAD_mapping, "parent1")
exampleRAD_mapping <- SetRecurrentParent(exampleRAD_mapping, "parent2")

# run the pipeline
exampleRAD_mapping <- PipelineMapping2Parents(exampleRAD_mapping)

# convert to polymapR format
examplePMR <- Export_polymapR(exampleRAD_mapping)

# export to MAPpoly
outfile2 <- tempfile() # temporary file for example
# generate a dummy phenotype matrix containing random numbers
mypheno <- matrix(rnorm(200), nrow = 100, ncol = 2,
                  dimnames = list(GetTaxa(exampleRAD_mapping)[-(1:2)],
                                  c("Height", "Yield")))
Export_MAPpoly(exampleRAD_mapping, file = outfile2, pheno = mypheno)

# load data into MAPpoly
# require(mappoly) # can uncomment once mappoly is on CRAN
# mydata <- read_geno_dist(outfile2)

# export to GWASpoly
outfile3 <- tempfile() # temporary file for example
Export_GWASpoly(exampleRAD, outfile3)
```

GetLikelyGen *Output the Most Likely Genotype*

Description

For a single taxon in a "RADdata" object, GetLikelyGen returns the most likely genotype (expressed in allele copy number) for each allele and each possible ploidy. The likelihoods used for determining genotypes are those stored in object$genotypeLikelihood.

Usage

```
GetLikelyGen(object, taxon, minLikelihoodRatio = 10)
```

Arguments

object	A "RADdata" object.
taxon	A character string indicating the taxon for which genotypes should be returned.
minLikelihoodRatio	A number indicating the minimum ratio of the likelihood of the most likely genotype to the likelihood of the second-most likely genotype for any genotype to be output for a given allele. If this number is one or less, all of the most likely genotypes will be output regardless of likelihood ratio. Where filtering is required so that only high confidence genotypes are retained, this number should be increased.

Value

A matrix with ploidies in rows (named with ploidies converted to character format) and alleles in columns. Each value indicates the most likely number of copies of that allele that the taxon has, assuming that ploidy.

Author(s)

Lindsay V. Clark

See Also

AddGenotypeLikelihood

Examples

```
# load dataset for this example
data(exampleRAD)
# add allele frequencies and genotype likelihoods
exampleRAD <- AddAlleleFreqHWE(exampleRAD)
exampleRAD <- AddGenotypeLikelihood(exampleRAD)

# get most likely genotypes
GetLikelyGen(exampleRAD, "sample001")
GetLikelyGen(exampleRAD, "sample002")

# try different filtering
GetLikelyGen(exampleRAD, "sample001", minLikelihoodRatio = 1)
GetLikelyGen(exampleRAD, "sample001", minLikelihoodRatio = 100)
```

GetWeightedMeanGenotypes *Export Numeric Genotype Values from Posterior Probabilities*

Description

These functions calculate numerical genotype values using posterior probabilities in a "RADdata" object, and output those values as a matrix of taxa by alleles. GetWeightedMeanGenotypes returns continuous genotype values, weighted by posterior genotype probabilities (*i.e.* posterior mean genotypes). GetProbableGenotypes returns discrete genotype values indicating the most probable genotype. If the "RADdata" object includes more than one possible inheritance mode, the $ploidyChiSq slot is used for selecting or weighting inheritance modes for each allele.

Usage

```
GetWeightedMeanGenotypes(object, ...)
## S3 method for class 'RADdata'
GetWeightedMeanGenotypes(object, minval = 0, maxval = 1,
                         omit1allelePerLocus = TRUE,
                         omitCommonAllele = TRUE,
                         naIfZeroReads = FALSE,
                         onePloidyPerAllele = FALSE, ...)

GetProbableGenotypes(object, ...)
## S3 method for class 'RADdata'
GetProbableGenotypes(object, omit1allelePerLocus = TRUE,
                     omitCommonAllele = TRUE,
                     naIfZeroReads = FALSE,
                     correctParentalGenos = TRUE, ...)
```

Arguments

object	A "RADdata" object. Posterior genotype probabilities should have been added with AddGenotypePosteriorProb, and if there is more than one possible ploidy, ploidy chi-squared values should have been added with AddPloidyChiSq.
...	Additional arguments, listed below, to be passed to the method for "RADdata".
minval	The number that should be used for indicating that a taxon has zero copies of an allele.
maxval	The number that should be used for indicating that a taxon has the maximum copies of an allele (equal to the ploidy of the locus).
omit1allelePerLocus	A logical indicating whether one allele per locus should be omitted from the output, in order to reduce the number of variables and prevent singularities for genome-wide association and genomic prediction. The value for one allele can be predicted from the values from all other alleles at its locus.

`omitCommonAllele`
: A logical, passed to the `commonAllele` argument of `OneAllelePerMarker`, indicating whether the most common allele for each locus should be omitted (as opposed to simply the first allele for each locus). Ignored if `omit1allelePerLocus = FALSE`.

`naIfZeroReads`
: A logical indicating whether `NA` should be inserted into the output matrix for any taxa and loci where the total read depth for the locus is zero. If `FALSE`, the output for these genotypes is essentially calculated using prior genotype probabilities, since prior and posterior genotype probabilities are equal when there are no reads.

`onePloidyPerAllele`
: Logical. If `TRUE`, for each allele the inheritance mode with the lowest χ^2 value is selected and is assumed to be the true inheritance mode. If `FALSE`, inheritance modes are weighted by inverse χ^2 values for each allele, and mean genotypes that have been weighted across inheritance modes are returned.

`correctParentalGenos`
: Logical. If `TRUE` and if the dataset was processed with `PipelineMapping2Parents`, the parental genotypes that are output are corrected according to the progeny allele frequencies, using the `likelyGeno_donor` and `likelyGeno_recurrent` slots in `object`. For the ploidy of the marker, the appropriate ploidy for the parents is selected using the `donorPloidies` and `recurrentPloidies` slots.

Details

For each inheritance mode m, taxon t, allele a, allele copy number i, total ploidy k, and posterior genotype probability $p_{i,t,a,m}$, posterior mean genotype $g_{t,a,m}$ is estimated by `GetWeightedMeanGenotypes` as:

$$g_{t,a,m} = \sum_{i=0}^{k} p_{i,t,a,m} * \frac{i}{k}$$

For `GetProbableGenotypes`, the genotype is the one with the maximum posterior probability:

$$g_{t,a,m} = i | \max_{i=0}^{k} p_{i,t,a,m}$$

When there are multiple inheritance modes and `onePloidyPerAllele = FALSE`, the weighted genotype is estimated by `GetWeightedMeanGenotypes` as:

$$g_{t,a} = \sum_{m} [g_{t,a,m} * \frac{1}{\chi^2_{m,a}} / \sum_{m} \frac{1}{\chi^2_{m,a}}]$$

In `GetProbableGenotypes`, or `GetWeightedMeanGenotypes` when there are multiple inheritance modes and `onePloidyPerAllele = TRUE`, the genotype is simply the one corresponding to the inheritance mode with the minimum χ^2 value:

$$g_{t,a} = g_{t,a,m} | \min_{m} \chi^2_{m,a}$$

Value

For `GetWeightedMeanGenotypes`, a named matrix, with taxa in rows and alleles in columns, and values ranging from `minval` to `maxval`. These values can be treated as continuous genotypes.

For `GetProbableGenotypes`, a list:

`genotypes`	A named integer matrix, with taxa in rows and alleles in columns, and values ranging from zero to the maximum ploidy for each allele. These values can be treated as discrete genotypes.
`ploidy_index`	A vector with one value per allele. It contains the index of the most inheritance mode of that allele in `object$priorProbPloidies`.

Author(s)

Lindsay V. Clark

Examples

```
# load dataset
data(exampleRAD_mapping)

# run a genotype calling pipeline;
# substitute with any pipeline and parameters
exampleRAD_mapping <- SetDonorParent(exampleRAD_mapping, "parent1")
exampleRAD_mapping <- SetRecurrentParent(exampleRAD_mapping, "parent2")
exampleRAD_mapping <- PipelineMapping2Parents(exampleRAD_mapping,
                                n.gen.backcrossing = 1, useLinkage = FALSE)


# get weighted mean genotypes
wmg <- GetWeightedMeanGenotypes(exampleRAD_mapping)
# examine the results
wmg[1:10,]

# get most probable genotypes
pg <- GetProbableGenotypes(exampleRAD_mapping, naIfZeroReads = TRUE)
# examine the results
pg$genotypes[1:10,]
```

`IterateHWE` *Iteratively Estimate Population Parameters and Genotypes In a Diversity Panel*

Description

These are wrapper function that iteratively run other **polyRAD** functions until allele frequencies stabilize to within a user-defined threshold. Genotype posterior probabilities can then be exported for downstream analysis.

Usage

```
IterateHWE(object, selfing.rate = 0, tol = 1e-05,
           excludeTaxa = GetBlankTaxa(object),
           overdispersion = 9)

IterateHWE_LD(object, selfing.rate = 0, tol = 1e-05,
              excludeTaxa = GetBlankTaxa(object),
              LDdist = 1e4, minLDcorr = 0.2,
              overdispersion = 9)

IteratePopStruct(object, selfing.rate = 0, tol = 1e-03,
                 excludeTaxa = GetBlankTaxa(object),
                 nPcsInit = 10, minfreq = 0.0001,
                 overdispersion = 9)

IteratePopStructLD(object, selfing.rate = 0, tol = 1e-03,
                   excludeTaxa = GetBlankTaxa(object),
                   nPcsInit = 10, minfreq = 0.0001, LDdist = 1e4,
                   minLDcorr = 0.2,
                   overdispersion = 9)
```

Arguments

object	A "RADdata" object.
selfing.rate	A number ranging from zero to one indicating the frequency of self-fertilization in the species.
tol	A number indicating when the iteration should end. It indicates the maximum mean difference in allele frequencies between iterations that is tolerated. Larger numbers will lead to fewer iterations.
excludeTaxa	A character vector indicating names of taxa that should be excluded from allele frequency estimates and chi-squared estimates.
nPcsInit	An integer indicating the number of principal component axes to initially estimate from object$depthRatio. Passed to AddPCA.
minfreq	A number indicating the minimum allele frequency allowed. Passed to AddAlleleFreqByTaxa.
LDdist	The distance, in basepairs, within which to search for alleles that may be in linkage disequilibrium with a given allele.
minLDcorr	The minimum correlation coefficient between two alleles for linkage disequilibrium between those alleles to be used by the pipeline for genotype estimation; see AddAlleleLinkages.
overdispersion	Overdispersion parameter; see AddGenotypeLikelihood.

Details

For IterateHWE, the following functions are run iteratively, assuming no population structure: AddAlleleFreqHWE, AddGenotypePriorProb_HWE, AddGenotypeLikelihood, AddPloidyChiSq, and AddGenotypePosteriorProb.

IterateHWE_LD runs each of the functions listed for IterateHWE once, then runs AddAlleleLinkages. It then runs AddAlleleFreqHWE, AddGenotypePriorProb_HWE, AddGenotypePriorProb_LD, AddGenotypeLikelihood, AddPloidyChiSq, and AddGenotypePosteriorProb iteratively until allele frequencies converge.

For IteratePopStruct, the following functions are run iteratively, modeling population structure: AddPCA, AddAlleleFreqByTaxa, AddAlleleFreqHWE, AddGenotypePriorProb_ByTaxa, AddGenotypeLikelihood, AddPloidyChiSq, and AddGenotypePosteriorProb. After the first PCA analysis, the number of principal component axes is not allowed to decrease, and can only increase by one from one round to the next, in order to help the algorithm converge.

IteratePopStructLD runs each of the functions listed for IteratePopStruct once, then runs AddAlleleLinkages. It then runs AddAlleleFreqHWE, AddGenotypePriorProb_ByTaxa, AddGenotypePriorProb_LD, AddGenotypeLikelihood, AddPloidyChiSq, AddGenotypePosteriorProb, AddPCA, and AddAlleleFreqByTaxa iteratively until convergence of allele frequencies.

Value

A "RADdata" object identical to that passed to the function, but with $alleleFreq, $priorProb, $genotypeLikelihood, $ploidyChiSq, and $posteriorProb slots added. For IteratePopStruct and IteratePopStructLD, $alleleFreqByTaxa and $PCA are also added. For IteratePopStructLD and IterateHWE_LD, $alleleLinkages and $priorProbLD are also added.

Note

If you see the error

```
Error in if (rel_ch < threshold & count > 5) { :   missing value where TRUE/FALSE needed
```

try lowering nPcsInit.

Author(s)

Lindsay V. Clark

See Also

GetWeightedMeanGenotypes for outputting genotypes in a useful format after iteration is completed.

StripDown to remove memory-hogging slots that are no longer needed after the pipeline has been run.

PipelineMapping2Parents for mapping populations.

Examples

```
# load dataset
data(exampleRAD)

# iteratively estimate parameters
exampleRAD <- IterateHWE(exampleRAD)

# export results
GetWeightedMeanGenotypes(exampleRAD)
```

```
# re-load to run pipeline assuming population structure
data(exampleRAD)

# run pipeline
exampleRAD <- IteratePopStruct(exampleRAD, nPcsInit = 3)

# export results
GetWeightedMeanGenotypes(exampleRAD)

# dataset for LD pipeline
data(Msi01genes)

# run HWE + LD pipeline
mydata1 <- IterateHWE_LD(Msi01genes)

# run pop. struct + LD pipeline
# (tolerance raised to make example run faster)
mydata2 <- IteratePopStructLD(Msi01genes, tol = 0.01)
```

LocusInfo *Get Information about a Single Locus*

Description

This function returns, and optionally prints, information about a single locus with a RADdata object, including alignment position, allele sequences, and genes overlapping the site.

Usage

```
LocusInfo(object, ...)
## S3 method for class 'RADdata'
LocusInfo(object, locus, genome = NULL,
          annotation = NULL, verbose = TRUE, ...)
```

Arguments

object	A RADdata object.
locus	A character string indicating the name of the locus to display. Alternatively, a character string indicating the name of an allele, for which the corresponding locus will be identified.
genome	An optional FaFile or BSgenome object containing the reference genome sequence.
annotation	An optional TxDb object containing the genome annotation.
verbose	If TRUE, results will be printed to the console.
...	Additional arguments (none implemented).

Details

The locus name, allele names, and allele sequences are always returned (although allele names are not printed with `verbose`). If the chromosome and position are known, those are also returned and printed. If `annotation` is provided, the function will return and print genes that overlap the locus. If `annotation` and `genome` are provided, the function will attempt to identify any amino acid changes caused by the alleles, using `predictCoding` internally. Identification of amino acid changes will work if the `RADdata` object was created with `VCF2RADdata` using the `refgenome` argument to fill in non-variable sites, and/or if the alleles are only one nucleotide long.

Value

A list containing:

`Locus`	The name of the locus.
`Chromosome`	The chromosome name, if present.
`Position`	The position in base pairs on the chromosome, if present.
`Alleles`	Allele names for the locus.
`Haplotypes`	Allele sequences for the locus, in the same order.
`Frequencies`	Allele frequencies, if present, in the same order.
`Transcripts`	Transcripts overlapping the locus, if an annotation was provided but it wasn't possible to predict amino acid changes.
`PredictCoding`	The output of `predictCoding`, if it was run.

Author(s)

Lindsay V. Clark

See Also

`makeTxDbFromGFF`, `GetLoci`

Examples

```
data(exampleRAD)
exampleRAD <- AddAlleleFreqHWE(exampleRAD)
loc2info <- LocusInfo(exampleRAD, "loc2")
```

`MakeTasselVcfFilter` *Filter Lines of a VCF File By Call Rate and Allele Frequency*

Description

This function creates another function that can be used as a prefilter by the function `filterVcf` in the package **VariantAnnotation**. The user can set a minimum number of indiviuals with reads and a minimum number of individuals with the minor allele (either the alternative or reference allele). The filter can be used to generate a smaller VCF file before reading with `VCF2RADdata`.

Usage

```
MakeTasselVcfFilter(min.ind.with.reads = 200, min.ind.with.minor.allele = 10)
```

Arguments

`min.ind.with.reads`
: An integer indicating the minimum number of individuals that must have reads in order for a marker to be retained.

`min.ind.with.minor.allele`
: An integer indicating the minimum number of individuals that must have the minor allele in order for a marker to be retained.

Details

This function assumes the VCF file was output by the TASSEL GBSv2 pipeline. This means that each genotype field begins with two digits ranging from zero to three separated by a forward slash to indicate the called genotype, followed by a colon.

Value

A function is returned. The function takes as its only argument a character vector representing a set of lines from a VCF file, with each line representing one SNP. The function returns a logical vector the same length as the character vector, with `TRUE` if the SNP meets the threshold for call rate and minor allele frequency, and `FALSE` if it does not.

Author(s)

Lindsay V. Clark

References

https://bitbucket.org/tasseladmin/tassel-5-source/wiki/Tassel5GBSv2Pipeline

Examples

```
# make the filtering function
filterfun <- MakeTasselVcfFilter(300, 15)


# Executable code excluded from CRAN testing for taking >10 s:

require(VariantAnnotation)
# get the example VCF installed with polyRAD
exampleVCF <- system.file("extdata", "Msi01genes.vcf", package = "polyRAD")
exampleBGZ <- paste(exampleVCF, "bgz", sep = ".")

# zip and index the file using Tabix (if not done already)
if(!file.exists(exampleBGZ)){
  exampleBGZ <- bgzip(exampleVCF)
  indexTabix(exampleBGZ, format = "vcf")
}

# filter to a new file
filterVcf(exampleBGZ, destination = "Msi01genes_filtered.vcf",
          prefilters = FilterRules(list(filterfun)))
```

MergeRareHaplotypes *Consolidate Reads from Rare Alleles*

Description

`MergeRareHaplotypes` searches for rare alleles in a `"RADdata"` object, and merges them into the most similar allele at the same locus based on nucleotide sequence (or the most common allele if multiple are equally similar). Read depth is summed across merged alleles, and the `alleleNucleotides` slot of the `"RADdata"` object contains IUPAC ambiguity codes to indicate nucleotide differences across merged alleles. This function is designed to be used immediately after data import.

Usage

```
MergeRareHaplotypes(object, ...)
## S3 method for class 'RADdata'
MergeRareHaplotypes(object, min.ind.with.haplotype = 10, ...)
```

Arguments

`object`	A `"RADdata"` object.
`min.ind.with.haplotype`	The minimum number of taxa having reads from a given allele for that allele to not be merged.
`...`	Additional arguments; none implemented.

Details

Alleles with zero reads across the entire dataset are removed by `MergeRareHaplotypes` without merging nucleotide sequences. After merging, at least one allele is left, even if it has fewer than `min.ind.with.haplotype` taxa with reads, as long as it has more than zero taxa with reads.

Value

A `"RADdata"` object identical to `object`, but with its `$alleleDepth`, `$antiAlleleDepth`, `$depthRatio`, `$depthSamplingPermutations`, `$alleleNucleotides`, and `$alleles2loc` arguments adjusted after merging alleles.

Author(s)

Lindsay V. Clark

See Also

`SubsetByLocus`, `VCF2RADdata`, `readStacks`

Examples

```
data(exampleRAD)
exampleRAD2 <- MergeRareHaplotypes(exampleRAD,
                                   min.ind.with.haplotype = 12)
exampleRAD$alleleDepth[21:30,3:5]
exampleRAD2$alleleDepth[21:30,3:4]
exampleRAD$alleleNucleotides
exampleRAD2$alleleNucleotides
```

MergeTaxaDepth *Combine Read Depths from Multiple Taxa into One Taxon*

Description

This function should be used in situations where data that were imported as separate taxa should be merged into a single taxon. The function should be used before any of the pipeline functions for genotype calling. Read depths are summed across duplicate taxa and output as a single taxon.

Usage

```
MergeTaxaDepth(object, ...)

## S3 method for class 'RADdata'
MergeTaxaDepth(object, taxa, ...)
```

Arguments

object	A RADdata object.
taxa	A character vector indicating taxa to be merged. The first taxon in the vector will be used to name the combined taxon in the output.
...	Additional arguments (none implemented).

Details

Examples of reasons to use this function:

- Duplicate samples across different libraries were given different names so that preliminary analysis could confirm that they were truly the same (*i.e.* no mix-ups) before combining them.
- Typos in the key file for the SNP mining software (TASSEL, Stacks, etc.) caused duplicate samples to have different names when they really should have had the same name.

To merge multiple sets of taxa into multiple combined taxa, this function can be run multiple times or in a loop.

Value

A RADdata object derived from object. The alleleDepth, antiAlleleDepth, locDepth, depthRatio, and depthSamplingPermutation slots, and "taxa" and "nTaxa" attributes, have been changed accordingly to reflect the merge.

Author(s)

Lindsay V. Clark

See Also

SubsetByTaxon

Examples

```
# dataset for this example
data(exampleRAD)

# merge the first three taxa into one
exampleRADm <- MergeTaxaDepth(exampleRAD, c("sample001", "sample002", "sample003"))

# inspect read depth
exampleRAD$alleleDepth[1:3,]
exampleRADm$alleleDepth[1:3,]
```

OneAllelePerMarker *Return the Index of One Allele for Each Locus*

Description

This function exists primarily to be called by functions such as AddPCA and GetWeightedMeanGenotypes that may need to exclude one allele per locus to avoid mathematical singularities. For a "RADdata" object, it returns the indices of one allele per locus.

Usage

```
OneAllelePerMarker(object, ...)
## S3 method for class 'RADdata'
OneAllelePerMarker(object, commonAllele = FALSE, ...)
```

Arguments

object	A "RADdata" object.
commonAllele	If TRUE, the index of the most common allele for each locus is returned, according to object$alleleFreq. If FALSE, the index of the first allele for each locus is returned.
...	Additional arguments (none implemented).

Value

An integer vector indicating the index of one allele for each locus in object.

Author(s)

Lindsay V. Clark

See Also

GetTaxa for a list of accessors.

Examples

```
data(exampleRAD)

OneAllelePerMarker(exampleRAD)

OneAllelePerMarker(exampleRAD, commonAllele = TRUE)
```

PipelineMapping2Parents *Run polyRAD Pipeline on a Mapping Population*

Description

This function is a wrapper for AddAlleleFreqMapping, AddGenotypeLikelihood, AddGenotypePriorProb_Mapping2Pare AddPloidyChiSq, and AddGenotypePosteriorProb. It covers the full pipeline for estimating genotype posterior probabilities from read depth in a "RADdata" object containing data from a mapping population.

Usage

```
PipelineMapping2Parents(object, n.gen.backcrossing = 0,
                        n.gen.intermating = 0, n.gen.selfing = 0,
                        donorParentPloidies = object$possiblePloidies,
                        recurrentParentPloidies = object$possiblePloidies,
                        minLikelihoodRatio = 10, freqAllowedDeviation = 0.05,
                        freqExcludeTaxa = c(GetDonorParent(object),
                                            GetRecurrentParent(object),
                                            GetBlankTaxa(object)),
                        useLinkage = TRUE, linkageDist = 1e7,
                        minLinkageCorr = 0.5, overdispersion = 9)
```

Arguments

object	A "RADdata" object.
n.gen.backcrossing	An integer, zero or greater, indicating how many generations of backcrossing to the recurrent parent were performed.
n.gen.intermating	An integer, zero or greater, indicating how many generations of intermating within the population were performed.
n.gen.selfing	An integer, zero or greater, indicating how many generations of selfing were performed.
donorParentPloidies	A list, where each item in the list is an integer vector indicating a potential inheritance mode that could be observed among loci in the donor parent. 2 indicates diploid, 4 indicates autotetraploid, c(2, 2) indicates, allotetraploid, *etc*.
recurrentParentPloidies	A list in the same format as donorParentPloidies indicating inheritance modes that could be observed among loci in the recurrent parent.
minLikelihoodRatio	The minimum likelihood ratio for determining parental genotypes with confidence, to be passed to GetLikelyGen for both parental taxa.

`freqAllowedDeviation`
: For `AddAlleleFreqMapping`, the amount by which an allele frequency can deviate from an expected allele frequency in order to be counted as that allele frequency.

`freqExcludeTaxa`
: A character vector indicating taxa to exclude from allele frequency estimates and ploidy χ^2 estimates.

`useLinkage`
: Boolean. Should genotypes at nearby loci (according to genomic alignment data) be used for updating genotype priors?

`linkageDist`
: A number, in basepairs, indicating the maximum distance for linked loci. Ignored if `useLinkage = FALSE`.

`minLinkageCorr`
: A number ranging from zero to one. Indicates the minimum correlation coeffienct between weighted mean genotypes at two alleles in order for linkage data to be used for updating genotype priors. Ignored if `useLinkage = FALSE`.

`overdispersion`
: Overdispersion parameter; see `AddGenotypeLikelihood`.

Details

Unlike `IterateHWE` and `IteratePopStruct`, `PipelineMapping2Parents` only runs through each function once, rather than iteratively until convergence.

Value

A `"RADdata"` object identical to that passed to the function, with the following slots added: `$alleleFreq`, `$genotypeLikelihood`, `$priorProb`, `$priorProbPloidies`, `$ploidyChiSq`, `$ploidyChiSqP`, and `$posteriorProb`. See the documentation for the functions listed in the description for more details on the data contained in these slots.

Author(s)

Lindsay V. Clark

See Also

`SetDonorParent` and `SetRecurrentParent` to indicate which individuals are the parents before running the function.

`GetWeightedMeanGenotypes` or `Export_polymapR` for exporting genotypes from the resulting object.

`StripDown` to remove memory-hogging slots that are no longer needed after the pipeline has been run.

Examples

```
# load data for the example
data(exampleRAD_mapping)

# specify donor and recurrent parents
exampleRAD_mapping <- SetDonorParent(exampleRAD_mapping, "parent1")
```

```
exampleRAD_mapping <- SetRecurrentParent(exampleRAD_mapping, "parent2")

# run the pipeline
exampleRAD_mapping <- PipelineMapping2Parents(exampleRAD_mapping,
                                  n.gen.backcrossing = 1)

# export results
wmgeno <- GetWeightedMeanGenotypes(exampleRAD_mapping)[-(1:2),]
wmgeno
```

`RADdata` *RADdata object constructor*

Description

`RADdata` is used internally to generate objects of the S3 class "RADdata" by **polyRAD** functions for importing read depth data. It is also available at the user level for cases where the data for import are not already in a format supported by **polyRAD**.

Usage

```
RADdata(alleleDepth, alleles2loc, locTable, possiblePloidies, contamRate,
        alleleNucleotides)

## S3 method for class 'RADdata'
plot(x, ...)
```

Arguments

`alleleDepth`	An integer matrix, with taxa in rows and alleles in columns. Taxa names should be included as row names. Each value indicates the number of reads for a given allele in a given taxon. There should be no `NA` values; use zero to indicate no reads.
`alleles2loc`	An integer vector with one value for each column of `alleleDepth`. The number indicates the identity of the locus to which the allele belongs. A locus can have any number of alleles assigned to it (including zero).
`locTable`	A data frame, where locus names are row names. There must be at least as many rows as the highest value of `alleles2loc`; each number in `alleles2loc` corresponds to a row index in `locTable`. No columns are required, although if provided a column named "Chr" will be used for indicating chromosome identities and a column named "Pos" will be used for indicating physical position.
`possiblePloidies`	A list, where each item in the list is an integer vector (or a numeric vector that can be converted to integer). Each vector indicates an inheritance pattern that SNPs in the dataset might obey. `2` indicates diploid, `4` indicates autotetraploid, `c(2, 2)` indicates allotetraploid, *etc.*

`contamRate` A number ranging from zero to one (although in practice probably less than 0.01) indicating the expected sample cross-contamination rate.

`alleleNucleotides`
A character vector with one value for each column of `alleleDepth`, indicating the DNA sequence for that allele. Typically only the sequence at variable sites is provided, although intervening non-variable sequence can also be provided.

`x` A "RADdata" object.

`...` Additional arguments to pass to `plot`, for example `col` or `pch`.

Value

An object of the S3 class "RADdata". The following slots are available using the `$` operator:

`alleleDepth` Identical to the argument provided to the function.

`alleles2loc` Identical to the argument provided to the function.

`locTable` Identical to the argument provided to the function.

`possiblePloidies`
The `possiblePloidies` argument, converted to integer.

`locDepth` A matrix with taxa in rows and loci in columns, with read depth summed across all alleles for each locus. Column names are locus numbers rather than locus names. See `GetLocDepth` for retrieving the same matrix but with locus names as column names.

`depthSamplingPermutations`
A numeric matrix with taxa in rows and alleles in columns. It is calculated as $log(locDepthchoosealleleDepth)$. This is used as a coefficient for likelihood estimations done by other **polyRAD** functions (*i.e.* `AddGenotypeLikelihood`).

`depthRatio` A numeric matrix with taxa in rows and alleles in columns. Calculated as $alleleDepth/locDepth$. Used by other **polyRAD** functions for rough estimation of genotypes and allele frequency.

`antiAlleleDepth`
An integer matrix with taxa in rows and alleles in columns. For each allele, the number of reads from the locus that do NOT belong to that allele. Calculated as $locDepth - alleleDepth$. Used for likelihood estimations by other **polyRAD** functions.

`alleleNucleotides`
Identical to the argument provided to the function.

The object additionally has several attributes (see `attr`):

`taxa` A character vector listing all taxa names, in the same order as the rows of `alleleDepth`.

`nTaxa` An integer indicating the number of taxa.

`nLoc` An integer indicating the number of loci in `locTable`.

`contamRate` Identical to the argument provided to the function.

The `plot` method performs a principal components analysis with `AddPCA` if not already done, then plots the first two axes. Points represent individuals (taxa). If mapping population parents have been noted in the object (see `SetDonorParent`), they are indicated in the plot.

Author(s)

Lindsay V. Clark

See Also

Data import functions that internally call RADdata:

readHMC, readTagDigger, VCF2RADdata, readStacks, readTASSELGBSv2

Examples

```
# create the dataset
mydepth <- matrix(sample(100, 16), nrow = 4, ncol = 4,
                  dimnames = list(paste("taxon", 1:4, sep = ""),
                  paste("loc", c(1,1,2,2), "_", c(0,1,0,1), sep = "")))
mydata <- RADdata(mydepth, c(1L,1L,2L,2L),
                  data.frame(row.names = c("loc1", "loc2"), Chr = c(1,1),
                             Pos = c(2000456, 5479880)),
                  list(2, c(2,2)), 0.001, c("A", "G", "G", "T"))

# inspect the dataset
mydata
mydata$alleleDepth
mydata$locDepth
mydata$depthRatio

# the S3 class structure is flexible; other data can be added
mydata$GPS <- data.frame(row.names = attr(mydata, "taxa"),
                         Lat = c(43.12, 43.40, 43.05, 43.27),
                         Long = -c(70.85, 70.77, 70.91, 70.95))
mydata$GPS

# If you have NA in your alleleDepth matrix to indicate zero reads,
# perform the following before running the RADdata constructor:
mydepth[is.na(mydepth)] <- 0L

# plotting a RADdata object
plot(mydata)
```

---

readHMC *Import read depth from UNEAK*

Description

This function reads the "HapMap.hmc.txt" and "HapMap.fas.txt" files output by the UNEAK pipeline and uses the data to generate a "RADdata" object.

Usage

```
readHMC(file, includeLoci = NULL, shortIndNames = TRUE,
        possiblePloidies = list(2), contamRate = 0.001,
        fastafile = sub("hmc.txt", "fas.txt", file, fixed = TRUE))
```

Arguments

`file`	Name of the file containing read depth (typically "HapMap.hmc.txt").
`includeLoci`	An optional character vector of loci to be included in the output.
`shortIndNames`	Boolean. If TRUE, taxa names will be shortened with respect to those in the file, eliminating all text after and including the first underscore.
`possiblePloidies`	A list of numeric vectors indicating potential inheritance modes of SNPs in the dataset. See `RADdata`.
`contamRate`	A number ranging from zero to one (typically small) indicating the expected rate of sample cross-contamination.
`fastafile`	Name of the file containing tag sequences (typically "HapMap.fas.txt").

Value

A `RADdata` object containing read depth, taxa and locus names, and nucleotides at variable sites.

Note

UNEAK is not able to report read depths greater than 127, which may be problematic for high depth data on polyploid organisms. The UNEAK pipeline is no longer being updated and is currently only available with archived versions of TASSEL.

Author(s)

Lindsay V. Clark

References

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.

http://maizegenetics.net/tassel

https://tassel.bitbucket.io/TasselArchived.html

See Also

`readTagDigger`, `VCF2RADdata`, `readStacks`, `readTASSELGBSv2`

Examples

```
# for this example we'll create dummy files rather than using real ones
hmc <- tempfile()
write.table(data.frame(rs = c("TP1", "TP2", "TP3"),
                       ind1_merged_X3 = c("15|0", "4|6", "13|0"),
                       ind2_merged_X3 = c("0|0", "0|1", "0|5"),
                       HetCount_allele1 = c(0, 1, 0),
                       HetCount_allele2 = c(0, 1, 0),
                       Count_allele1 = c(15, 4, 13),
                       Count_allele2 = c(0, 7, 5),
                       Frequency = c(0, 0.75, 0.5)), row.names = FALSE,
            quote = FALSE, col.names = TRUE, sep = "\t", file = hmc)
fas <- tempfile()
writeLines(c(">TP1_query_64",
             "TGCAGAAAAAAAACGCTCGATGCCCCCTAATCCGTTTTCCCCATTCCGCTCGCCCCATCGGAGT",
             ">TP1_hit_64",
             "TGCAGAAAAAAAACGCTCGATGCCCCCTAATCCGTTTTCCCCATTCCGCTCGCCCCATTGGAGT",
             ">TP2_query_64",
             "TGCAGAAAAACAACACCCTAGGTAACAACCATATCTTATATTGCCGAATAAAAAACAACACCCC",
             ">TP2_hit_64",
             "TGCAGAAAAACAACACCCTAGGTAACAACCATATCTTATATTGCCGAATAAAAAATAACACCCC",
             ">TP3_query_64",
             "TGCAGAAAACATGGAGAGGGAGATGGCACGGCAGCACCACCGCTGGTCCGCTGCCCGTTTGCGG",
             ">TP3_hit_64",
             "TGCAGAAAACATGGAGATGGAGATGGCACGGCAGCACCACCGCTGGTCCGCTGCCCGTTTGCGG"),
             fas)

# now read the data
mydata <- readHMC(hmc, fastafile = fas)

# inspect the results
mydata
mydata$alleleDepth
mydata$alleleNucleotides
row.names(mydata$locTable)
```

---

readStacks *Import Read Depth from Stacks*

Description

Using the catalog files output by cstacks and matches file output by sstacks, this function imports read depth into a RADdata object. If genomic alignments were used, alignment data can optionally be imported.

Usage

```
readStacks(allelesFile, matchesFolder, version = 2,
           min.ind.with.reads = 200,
```

```
min.ind.with.minor.allele = 10, readAlignmentData = FALSE,
sumstatsFile = "populations.sumstats.tsv",
possiblePloidies = list(2), contamRate = 0.001)
```

Arguments

`allelesFile`	Path to the "alleles" file from the Stacks catalog.
`matchesFolder`	Path to the folder containing "matches" files to import.
`version`	Either the number 1 or 2, indicating the version of Stacks.
`min.ind.with.reads`	For filtering loci. A locus must have at least this many samples with reads in order to be retained.
`min.ind.with.minor.allele`	For filtering loci. A locus must have at least this many samples with reads for the minor allele in order to be retained. For loci with more than two alleles, at least two alleles must be present in at least this many individuals. This argument is also passed internally to the `min.ind.with.haplotype` argument of `MergeRareHaplotypes` to consolidate reads from rare alleles.
`readAlignmentData`	If `TRUE` and `version = 1`, the "tags" file from the Stacks catalog will be read, and chromosome, position, and strand will be imported to the `locTable` slot of the output. It is assumed that the "tags" file is in the same directory as the "alleles" file. If `TRUE` and `version = 2`, `sumstatsFile` will be used for import of chromosome and position data.
`sumstatsFile`	The name of the file containing summary statistics for loci. Ignored unless `version = 2` and `readAlignmentData = TRUE`.
`possiblePloidies`	A list indicating possible inheritance modes in the dataset. See `RADdata`.
`contamRate`	A number from 0 to 1 (generally very small) indicating the expected rate of cross contamination between samples.

Value

A `RADdata` object.

Note

This function has been tested with output from Stacks 1.47.

Author(s)

Lindsay V. Clark

References

Stacks website: http://catchenlab.life.illinois.edu/stacks/

Rochette, N. and Catchen, J. (2017) Deriving genotypes from RAD-seq short-read data using Stacks. *Nature Protocols* **12**, 2640–2659.

Catchen, J., Hohenlohe, P. A., Bassham, S., Amores, A., and Cresko., W. A. (2013) Stacks: an analysis tool set for population genomics. *Molecular Ecology* **22**, 3124–3140.

Catchen, J. M., Amores, A., Hohenlohe, P., Cresko, W., and Postlethwait, J. H. (2011) Stacks: building and genotyping loci de novo from short-read sequences. *G3: Genes, Genomes, Genetics* **1**, 171–182.

See Also

VCF2RADdata, readTagDigger, readHMC, readTASSELGBSv2

Examples

```
## Not run:

# Assuming the working directory contains the catalog and all matches files:

myStacks <- readStacks("batch_1.catalog.alleles.tsv", ".",
                       version = 1,
                       readAlignmentData = TRUE)

## End(Not run)
```

---

readTagDigger *Import Read Counts from TagDigger*

Description

readTagDigger reads the CSV output containing read counts from TagDigger and generates a "RADdata" object. Optionally, it can also import a tag database generated by the Tag Manager program within TagDigger, containing information such as alignment position, to be stored in the $locTable slot of the "RADdata" object.

Usage

```
readTagDigger(countfile, includeLoci = NULL,
              possiblePloidies = list(2), contamRate = 0.001,
              dbfile = NULL, dbColumnsToKeep = NULL,
              dbChrCol = "Chr", dbPosCol = "Pos",
              dbNameCol = "Marker name")
```

Arguments

countfile	Name of the file containing read counts.
includeLoci	An optional character vector containing names of loci to retain in the output.
possiblePloidies	A list of numeric vectors indicating potential inheritance modes of SNPs in the dataset. See RADdata.

contamRate	A number ranging from zero to one (typically small) indicating the expected rate of sample cross-contamination.
dbfile	Optionally, name of the Tag Manager database file.
dbColumnsToKeep	Optionally, a character vector indicating the names of columns to keep from the database file.
dbChrCol	The name of the column containing the chromosome number in the database file.
dbPosCol	The name of the column indicating alignment position in the database file.
dbNameCol	The name of the column containing marker names in the database file.

Details

Nucleotides associated with the alleles, to be stored in the `$alleleNucleotides` slot, are extracted from the allele names in the read counts file. It is assumed that the allele names first contain the marker name, followed by an underscore, followed by the nucleotide(s) at any variable positions.

Value

A "RADdata" object.

Author(s)

Lindsay V. Clark

References

https://github.com/lvclark/tagdigger

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**, 11.

See Also

readHMC, readStacks, VCF2RADdata, readTASSELGBSv2

Examples

```
# for this example we'll create dummy files
countfile <- tempfile()
write.csv(data.frame(row.names = c("Sample1", "Sample2", "Sample3"),
                     Mrkr1_A_0 = c(0, 20, 4),
                     Mrkr1_G_1 = c(7, 0, 12)),
          file = countfile, quote = FALSE)
dbfile <- tempfile()
write.csv(data.frame(Marker.name = "Mrkr1", Chr = 5, Pos = 66739827),
          file = dbfile, row.names = FALSE, quote = FALSE)

# read the data
myrad <- readTagDigger(countfile, dbfile = dbfile)
```

readTASSELGBSv2 *Import Read Depth and Alignment from TASSEL GBS v2*

Description

This function reads TagTaxaDist and SAM files output by the TASSEL 5 GBS v2 pipeline, and generates a RADdata object suitable for downstream processing for genotype estimation. It elimintes the need to run the DiscoverySNPCallerPluginV2 or the ProductionSNPCallerPluginV2, since **polyRAD** operates on haplotypes rather than SNPs.

Usage

```
readTASSELGBSv2(tagtaxadistFile, samFile, min.ind.with.reads = 200,
                min.ind.with.minor.allele = 10, possiblePloidies = list(2),
                contamRate = 0.001, chromosomes = NULL)
```

Arguments

tagtaxadistFile	File name or path to a tab-delimited text file of read depth generated by the GetTagTaxaDistFromDBPlugin in TASSEL.
samFile	File name or path to the corresponding SAM file containing alignment information for the same set of tags. This file is obtained by running the TagExportToFastqPlugin in TASSEL, followed by alignment using Bowtie2 or BWA.
min.ind.with.reads	Integer used for marker filtering. The minimum number of individuals that must have read depth above zero for a locus to be retained in the output.
min.ind.with.minor.allele	Integer used for marker filtering. The minimum number of individuals possessing reads for the minor allele for a locus to be retained in the output. This value is also passed to the min.ind.with.haplotype argument of MergeRareHaplotypes.
possiblePloidies	A list indicating inheritance modes that might be encountered in the dataset. See RADdata.
contamRate	A number indicating the expected sample cross-contamination rate. See RADdata.
chromosomes	A character vector of chromosome names, indicating chromosomes to be retained in the output. If NULL, all chromosomes to be retained. This argument is intended to be used for reading data in a chromosome-wise fashion in order to conserve computer memory.

Value

A RADdata object containing read depth and alignment infomation from the two input files.

Note

Sequence tags must be identical in length to be assigned to the same locus by this function. This is to prevent errors with MergeRareHaplotypes.

Author(s)

Lindsay V. Clark

References

TASSEL GBSv2 pipeline: https://bitbucket.org/tasseladmin/tassel-5-source/wiki/Tassel5GBSv2Pipeline

Bowtie2: http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

BWA: http://bio-bwa.sourceforge.net/

See Also

Other data import functions: readStacks, readHMC, readTagDigger, VCF2RADdata

Examples

```
# get files for this example
samfile <- system.file("extdata", "exampleTASSEL_SAM.txt",
                       package = "polyRAD")
ttdfile <- system.file("extdata", "example_TagTaxaDist.txt",
                       package = "polyRAD")

# import data
myrad <- readTASSELGBSv2(ttdfile, samfile, min.ind.with.reads = 8,
                         min.ind.with.minor.allele = 2)
```

SetBlankTaxa *Functions to Assign Taxa to Specific Roles*

Description

These functions are used for assigning and retrieving taxa from a "RADdata" object that serve particular roles in the dataset. Blank taxa can be used for estimating the contamination rate (see EstimateContaminationRate), and the donor and recurrent parents are used for determining expected genotype distributions in mapping populations. Many functions in **polyRAD** will automatically exclude taxa from analysis if they have been assigned to one of these roles.

Usage

```
SetBlankTaxa(object, value)
GetBlankTaxa(object, ...)
SetDonorParent(object, value)
GetDonorParent(object, ...)
SetRecurrentParent(object, value)
GetRecurrentParent(object, ...)
```

Arguments

object	A "RADdata" object.
value	A character string (or a character vector for SetBlankTaxa) indicating the taxon or taxa to be assigned to the role.
...	Other arguments (none currently supported).

Value

For the "Get" functions, a character vector indicating the taxon or taxa that have been assigned to that role. For the "Set" functions, a "RADdata" object identical to the one passed to the function, but with new taxa assigned to that role.

Author(s)

Lindsay V. Clark

See Also

AddGenotypePriorProb_Mapping2Parents

Examples

```
# assign parents in a mapping population
data(exampleRAD_mapping)
exampleRAD_mapping <- SetDonorParent(exampleRAD_mapping, "parent1")
exampleRAD_mapping <- SetRecurrentParent(exampleRAD_mapping, "parent2")
GetDonorParent(exampleRAD_mapping)
GetRecurrentParent(exampleRAD_mapping)

# assign blanks
exampleRAD_mapping <- SetBlankTaxa(exampleRAD_mapping,
                                   c("progeny019", "progeny035"))
GetBlankTaxa(exampleRAD_mapping)
```

StripDown *Remove Unneeded Slots to Conserve Memory*

Description

This function is designed to be used after a RADdata object has been processed by one of the pipeline functions. Slots that are no longer needed are removed in order to conserve memory.

Usage

```
StripDown(object, ...)
## S3 method for class 'RADdata'
StripDown(object,
          remove.slots = c("depthSamplingPermutations",
                           "depthRatio", "antiAlleleDepth",
                           "genotypeLikelihood", "priorProb",
                           "priorProbLD"),
          ...)
```

Arguments

`object`	A `RADdata` object.
`remove.slots`	A character vector listing slots that will be removed.
`...`	Additional arguments (none implemented).

Details

The default slots that are removed take up a lot of memory but are not used by the export functions. Other slots to consider removing are `alleleFreq`, `alleleFreqByTaxa`, `PCA`, `locDepth`, `alleleDepth`, and `alleleLinkages`. Of course, if you have custom uses for some of the slots that are removed by default, you can change the `remove.slots` vector to not include them.

The function will throw an error if the user attempts to remove key slots that are needed for export and downstream analysis, including:

- alleles2loc
- alleleNucleotides
- locTable
- priorProbPloidies
- possiblePloidies
- ploidyChiSq
- posteriorProb

Value

A `RADdata` object

Author(s)

Lindsay V. Clark

See Also

`SubsetByTaxon`, `SubsetByLocus`

Examples

```
# load a dataset for this example
data(exampleRAD)

# run a pipeline
exampleRAD <- IterateHWE(exampleRAD)

# check the size of the resulting object
object.size(exampleRAD)

# remove unneeded slots
exampleRAD <- StripDown(exampleRAD)

# check object size again
object.size(exampleRAD)
```

SubsetByLocus *Create RADdata Objects with a Subset of Loci*

Description

These functions take a RADdata object as input and generate smaller RADdata objects containing only the specified loci. SubsetByLocus allows the user to specify which loci are kept, whereas SplitByChromosome creates multiple RADdata objects representing chromosomes or sets of chromosomes. RemoveMonomorphicLoci eliminates any loci with fewer than two alleles. RemoveHighDepthLoci eliminates loci that have especially high read depth in order to eliminate false loci originating from repetitive sequence. RemoveUngenotypedLoci is intended for datasets that have been run through PipelineMapping2Parents and may have some genotypes that are missing or non-variable due to how priors were determined.

Usage

```
SubsetByLocus(object, ...)
## S3 method for class 'RADdata'
SubsetByLocus(object, loci, ...)

SplitByChromosome(object, ...)
## S3 method for class 'RADdata'
SplitByChromosome(object, chromlist = NULL, chromlist.use.regex = FALSE,
                  fileprefix = "splitRADdata", ...)

RemoveMonomorphicLoci(object, ...)
## S3 method for class 'RADdata'
RemoveMonomorphicLoci(object, verbose = TRUE, ...)

RemoveHighDepthLoci(object, ...)
## S3 method for class 'RADdata'
RemoveHighDepthLoci(object, max.SD.above.mean = 2, verbose = TRUE, ...)
```

```
RemoveUngenotypedLoci(object, ...)
## S3 method for class 'RADdata'
RemoveUngenotypedLoci(object, removeNonvariant = TRUE, ...)
```

Arguments

object	A RADdata object.
loci	A character or numeric vector indicating which loci to include in the output RADdata object. If numeric, it refers to row numbers in object$locTable. If character, it refers to row names in object$locTable.
chromlist	An optional list indicating how chromosomes should be split into separate RADdata objects. Each item in the list is a vector of the same class as object$locTable$Chr (character or numeric) containing the names of chromosomes that should go into one group. If not provided, each chromosome will be sent to a separate RADdata object.
chromlist.use.regex	If TRUE, the character strings in chromlist will be treated as regular expressions for searching chromosome names. For example, if one wanted all chromosomes beginning with the string "scaffold" to go into one RADdata object, one could include the string "^scaffold" as an item in chromlist and set chromlist.use.regex = TRUE. If FALSE, exact matches to chromosome names will be used.
fileprefix	A character string indicating the prefix of .RData files to export.
max.SD.above.mean	The maximum number of standard deviations above the mean read depth that a locus can be in order to be retained.
verbose	If TRUE, print out information about the original number of loci and the number of loci that were retained. For RemoveHighDepthLoci, a histogram is also plotted showing mean depth per locus, and the cutoff for removing loci.
removeNonvariant	If TRUE, in addition to removing loci where posterior probabilities are missing, loci will be removed where posterior probabilities are uniform across the population.
...	Additional arguments (none implemented).

Details

SubsetByLocus may be useful if the user has used their own filtering criteria to determine a set of loci to retain, and wants to create a new dataset with only those loci. It can be used at any point in the analysis process.

SplitByChromosome is intended to make large datasets more manageable by breaking them into smaller datasets that can be processed independently, either in parallel computing jobs on a cluster, or one after another on a computer with limited RAM. Generally it should be used immediately after data import. Rather than returning new RADdata objects, it saves them individually to separate workspace image files, which can than be loaded one at a time to run analysis pipelines such as

IteratePopStruct. GetWeightedMeanGenotypes or one of the export functions can be run on each resulting RADdata object, and the resulting matrices concatenated with cbind.

SplitByChromosome, RemoveMonomorphicLoci, and RemoveHighDepthLoci use SubsetByLocus internally.

Value

SubsetByLocus, RemoveMonomorphicLoci, RemoveHighDepthLoci, and RemoveUngenotypedLoci return a RADdata object with all the slots and attributes of object, but only containing the loci listed in loci, only loci with two or more alleles, only loci without abnormally high depth, or only loci where posterior probabilities are non-missing and variable, respectively.

SplitByChromosome returns a character vector containing file names where .RData files have been saved. Each .RData file contains one RADdata object named splitRADdata.

Author(s)

Lindsay V. Clark

See Also

VCF2RADdata, SubsetByTaxon

Examples

```
# load a dataset for this example
data(exampleRAD)
exampleRAD

# just keep the first and fourth locus
subsetRAD <- SubsetByLocus(exampleRAD, c(1, 4))
subsetRAD

# split by groups of chromosomes
exampleRAD$locTable
tf <- tempfile()
splitfiles <- SplitByChromosome(exampleRAD, list(c(1, 4), c(6, 9)),
                                fileprefix = tf)
load(splitfiles[1])
splitRADdata

# filter out monomorphic loci (none removed in example)
filterRAD <- RemoveMonomorphicLoci(exampleRAD)

# filter out high depth loci (none removed in this example)
filterRAD2 <- RemoveHighDepthLoci(filterRAD)

# filter out loci with missing or non-variable genotypes
# (none removed in this example)
filterRAD3 <- IterateHWE(filterRAD2)
filterRAD3 <- RemoveUngenotypedLoci(filterRAD3)
```

SubsetByPloidy *Create a RADdata object with a Subset of Possible Ploidies*

Description

This function is used for removing some of the ploidies (inheritance modes) stored in a RADdata object. If genotype calling has already been performed, all of the relevant slots will be subsetted to only keep the ploidies that the user indicates.

Usage

```
SubsetByPloidy(object, ...)
## S3 method for class 'RADdata'
SubsetByPloidy(object, ploidies, ...)
```

Arguments

object	A RADdata object.
ploidies	A list, formatted like object$possiblePloidies, indicating ploidies to retain. Each item in the list is a vector, where 2 indicates diploid, c(2, 2) allotetraploid, 4 autotetraploid, etc.
...	Other arguments (none implemented).

Details

Note that slots of object are subsetted but not recalculated. For example, GetWeightedMeanGenotypes takes a weighted mean across ploidies, which is in turn used for estimating allele frequencies and performing PCA. If the values in object$ploidyChiSq are considerably higher for the ploidies being removed than for the ploidies being retained, this difference is likely to be small and not substantially impact genotype calling. Otherwise, it may be advisable to re-run genotype calling after running SubsetByPloidy.

Value

A RADdata object identical to object, but only containing data relevant to the inheritance modes listed in ploidies.

Author(s)

Lindsay V. Clark

See Also

SubsetByTaxon, SubsetByLocus

Examples

```
# Example dataset assuming diploidy or autotetraploidy
data(exampleRAD)
exampleRAD <- IterateHWE(exampleRAD)
# Subset to only keep tetraploid results
exampleRAD <- SubsetByPloidy(exampleRAD, ploidies = list(4))
```

SubsetByTaxon *Create RADdata Object with a Subset of Taxa*

Description

This function is used for removing some of the taxa from a dataset stored in a RADdata object.

Usage

```
SubsetByTaxon(object, ...)
## S3 method for class 'RADdata'
SubsetByTaxon(object, taxa, ...)
```

Arguments

object	A RADdata object.
taxa	A character or numeric vector indicating which taxa to retain in the output.
...	Additional arguments (none implemented).

Details

This function may be used for subsetting a RADdata object either immediately after data import, or after additional analysis has been performed. Note however that estimation of allele frequencies, genotype prior probabilities, PCA, *etc.* are very dependent on what samples are included in the dataset. If those calculations have already been performed, the results will be transferred to the new object but not recalculated.

Value

A RADdata object containing only the taxa listed in taxa.

Author(s)

Lindsay V. Clark

See Also

SubsetByLocus

Examples

```
# load data for this example
data(exampleRAD)
exampleRAD

# just keep the first fifty taxa
subsetRAD <- SubsetByTaxon(exampleRAD, 1:50)
subsetRAD
```

TestOverdispersion *Test the Fit of Read Depth to Beta-Binomial Distribution*

Description

This function is intended to help the user select a value to pass to the `overdispersion` argument of `AddGenotypeLikelihood`, generally via pipeline functions such as `IterateHWE` or `PipelineMapping2Parents`.

Usage

```
TestOverdispersion(object, ...)

## S3 method for class 'RADdata'
TestOverdispersion(object, to_test = seq(6, 20, by = 2), ...)
```

Arguments

`object`	A `RADdata` object. Genotype calling does not need to have been performed, although for mapping populations it might be helpful to have done a preliminary run of `PipelineMapping2Parents` without linkage.
`to_test`	A vector containing values to test. These are values that will potentially be used for the `overdispersion` argument of a pipeline function. They should all be positive numbers.
`...`	Additional arguments (none implemented).

Details

If no genotype calling has been performed, a single iteration under HWE using default parameters will be done. `object$ploidyChiSq` is then examined to determine the most common/most likely inheritance mode for the whole dataset. The alleles that are examined are only those where this inheritance mode has the lowest chi-squared value.

Within this inheritance mode and allele set, genotypes are selected where the posterior probability of having a single copy of the allele is at least 0.95. Read depth for these genotypes is then analyzed. For each genotype, a two-tailed probability is calculated for the read depth ratio to deviate from the expected ratio by at least that much under the beta-binomial distribution. This test is performed for each overdispersion value provided in `to_test`.

Value

A list of the same length as `to_test`. The names of the list are `to_test` converted to a character vector. Each item in the list is a vector of p-values, one per examined genotype, of the read depth ratio for that genotype to deviate that much from the expected ratio.

Author(s)

Lindsay V. Clark

Examples

```
# dataset with overdispersion
data(Msi01genes)

# test several values for the overdispersion parameter
myP <- TestOverdispersion(Msi01genes, to_test = 8:10)

# visualize results with QQ plots
require(qqman)
qq(myP[["8"]])  # over-fit; too much overdispersion in model
qq(myP[["9"]])  # fairly close to expected; good value to use
qq(myP[["10"]]) # slightly under-fit; not enough overdispersion
```

VCF2RADdata *Create a RADdata Object from a VCF File*

Description

This function reads a Variant Call Format (VCF) file containing allelic read depth and SNP alignment positions, such as can be produced by TASSEL or GATK, and generates a `RADdata` dataset to be used for genotype calling in **polyRAD**.

Usage

```
VCF2RADdata(file, phaseSNPs = TRUE, tagsize = 80, refgenome = NULL,
            tol = 0.01, al.depth.field = "AD", min.ind.with.reads = 200,
            min.ind.with.minor.allele = 10, possiblePloidies = list(2),
            contamRate = 0.001,
          samples = VariantAnnotation::samples(VariantAnnotation::scanVcfHeader(file)),
            svparam = VariantAnnotation::ScanVcfParam(fixed = "ALT", info = NA,
                                                      geno = al.depth.field,
                                                      samples = samples),
            yieldSize = 5000, expectedAlleles = 5e+05, expectedLoci = 1e+05,
            maxLoci = NA)
```

Arguments

file	The path to a VCF file to be read. This can be uncompressed, bgzipped using Samtools or Bioconductor, or a TabixFile object from Bioconductor.
phaseSNPs	If TRUE, markers that appear to have come from the same set of reads will be phased and grouped into haplotypes. Otherwise, each row of the file will be kept as a distinct marker.
tagsize	The read length, minus any barcode sequence, that was used for genotyping. In TASSEL, this is the same as the kmerLength option. This argument is used for grouping SNPs into haplotypes and is ignored if phaseSNPs = FALSE.
refgenome	Optional. The name of a FASTA file, or an FaFile object, containing the reference genome. When grouping SNPs into haplotypes, if provided this reference genome is used to insert non-variable nucleotides between the variable nucleotides in the alleleNucleotides slot of the RADdata output. Ignored if phaseSNPs = FALSE. Useful if exact SNP positions need to be retained for downstream analysis after genotype calling in **polyRAD**.
tol	The proportion by which two SNPs can differ in read depth and still be merged into one group for phasing. Ignored if phaseSNPs = FALSE.
al.depth.field	The name of the genotype field in the VCF file that contains read depth at each allele. This should be "AD" unless your format is very unusual.
min.ind.with.reads	Integer used for filtering SNPs. To be retained, a SNP must have at least this many samples with reads.
min.ind.with.minor.allele	Integer used for filtering SNPs. To be retained, a SNP must have at least this many samples with the minor allele. When there are more than two alleles, at least two alleles must have at least this many samples with reads for the SNP to be retained.
possiblePloidies	A list indicating inheritance modes that might be encountered in the dataset. See RADdata.
contamRate	A number indicating the expected sample cross-contamination rate. See RADdata.
samples	A character vector containing the names of samples from the file to export to the RADdata object. The default is all samples. If a subset is provided, filtering with min.ind.with.reads and min.ind.with.minor.allele is performed within that subset. Ignored if a different samples argument is provided within svparam.
svparam	A ScanVcfParam object to be used with readVcf. The primary reasons to change this from the default would be 1) if you want additional FIXED or INFO fields from the file to be exported to the locTable slot of the RADdata object, and/or 2) if you only want to import particular regions of the genome, as specified with the which argument of ScanVcfParam.
yieldSize	An integer indicating the number of lines of the file to read at once. Increasing this number will make the function faster but consume more RAM.
expectedAlleles	An integer indicating the approximate number of alleles that are expected to be imported after filtering and phasing. If this number is too low, the function may

slow down considerably. Increasing this number increases the amount of RAM used by the function.

`expectedLoci` An integer indicating the approximate number of loci that are expected to be imported after filtering and phasing. If this number is too low, the function may slow down considerably. Increasing this number increases the amount of RAM used by the function.

`maxLoci` An integer indicating the approximate maximum number of loci to return. If provided, the function will stop reading the file once it has found at least this many loci that pass filtering and phasing. This argument is intended to be used for generating small `RADdata` objects for testing purposes, and should be left `NA` under normal circumstances.

Details

This function requires the BioConductor package **VariantAnnotation**. See https://bioconductor.org/packages/release/bioc/html/VariantAnnotation.html for installation instructions.

If you anticipate running `VCF2RADdata` on the same file more than once, it is recommended to run `bgzip` and `indexTabix` from the package **Rsamtools** once before running `VCF2RADdata`. See examples.

`min.ind.with.minor.allele` is used for filtering SNPs as the VCF file is read. Additionally, because phasing SNPs into haplotypes can cause some haplotypes to fail to pass this threshold, `VCF2RADdata` internally runs `MergeRareHaplotypes` with `min.ind.with.haplotype = min.ind.with.minor.allele`, then `RemoveMonomorphicLoci`, before returning the final `RADdata` object.

Value

A `RADdata` object.

Author(s)

Lindsay V. Clark

References

Variant Call Format specification: http://samtools.github.io/hts-specs/

TASSEL GBSv2 pipeline: https://bitbucket.org/tasseladmin/tassel-5-source/wiki/Tassel5GBSv2Pipeline

GATK: https://software.broadinstitute.org/gatk/

Tassel4-Poly: https://github.com/guilherme-pereira/tassel4-poly

See Also

`MakeTasselVcfFilter` for filtering to a smaller VCF file before reading with `VCF2RADdata`.

Other data import functions: `readStacks`, `readHMC`, `readTagDigger`, `readTASSELGBSv2`

Examples

```
# get the example VCF installed with polyRAD
exampleVCF <- system.file("extdata", "Msi01genes.vcf", package = "polyRAD")


# loading VariantAnnotation namespace takes >10s,
# so is excluded from CRAN checks

require(VariantAnnotation)

# Compress and index the VCF before reading, if not already done
if(!file.exists(paste(exampleVCF, "bgz", sep = "."))){
  vcfBG <- bgzip(exampleVCF)
  indexTabix(vcfBG, "vcf")
}

# Read into RADdata object
myRAD <- VCF2RADdata(exampleVCF, expectedLoci = 100, expectedAlleles = 500)

# Example of subsetting by genomic region (first 200 kb on Chr01)
mysv <- ScanVcfParam(fixed = "ALT", info = NA, geno = "AD",
                     samples = samples(scanVcfHeader(exampleVCF)),
                     which = GRanges("01", IRanges(1, 200000)))
myRAD2 <- VCF2RADdata(exampleVCF, expectedLoci = 100, expectedAlleles = 500,
                      svparam = mysv, yieldSize = NA_integer_)
```

Index