Package ‘updog’

July 27, 2018

Title Flexible Genotyping for Polyploids

Version 1.0.1

Date 2018-06-11

Description Implements empirical Bayes approaches to genotype polyploids from next generation sequencing data while accounting for allelic bias, overdispersion, and sequencing error. The main function is flexdog(), which allows the specification of many different genotype distributions. An experimental function that takes into account varying levels of relatedness is implemented in mupdog(). Also provided are functions to simulate genotypes, rgeno(), and read-counts, rflexdog(), as well as functions to calculate oracle genotyping error rates, oracle_mis(), and correlation with the true genotypes, oracle_cor(). These latter two functions are useful for read depth calculations. Run browseVignettes(package = ``updog'') in R for example usage. See also Gerard et al. (2018) <doi:10.1101/281550> for details on the implemented methods.

Depends R (>= 3.4.0)

License GPL-3

BugReports `http://github.com/dcgerard/updog/issues`

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp (>= 0.12.16), RcppArmadillo, assertthat, foreach, doParallel, ggplot2, ggthemes, stringr

Suggests covr, testthat, SuppDists, Rmpfr, CVXR, knitr, rmarkdown, tidyverse

VignetteBuilder knitr

NeedsCompilation yes

Author David Gerard [aut, cre] (<https://orcid.org/0000-0001-9450-5023>)

Maintainer David Gerard <gerard.1787@gmail.com>

Repository CRAN

Date/Publication 2018-07-27 17:50:06 UTC

R topics documented:

`ashpen_fun` *Penalty on pivec used when* `model = "ash"` *in* `flexdog`.

Description

Penalty on pivec used when `model = "ash"` in `flexdog`.

Usage

```
ashpen_fun(lambda, pivec)
```

Arguments

`lambda`	The penalty.
`pivec`	The vector of mixing proportions for the component discrete uniform distributions.

Value

A penalty on the ash mixing weights.

Author(s)

David Gerard

`compute_all_log_bb` *Calculates the log-density for every individual by snp by dosage level.*

Description

Calculates the log-density for every individual by snp by dosage level.

Usage

```
compute_all_log_bb(refmat, sizemat, ploidy, seq, bias, od)
```

Arguments

`refmat`	A matrix of reference counts. The rows index the individuals and the columns index the SNPs.
`sizemat`	A matrix of total counts. The rows index the individuals and the columns index the SNPs. Should have the same dimensions as `refmat`.
`ploidy`	The ploidy of the species. To estimate the ploidy, re-run `mupdog` at various ploidy levels and choose the one with the largest ELBO. This assumes that the ploidy is the same for all individuals in the sample.
`seq`	A vector of initial sequencing errors. Should be the same length as the number of columns of `refmat` (number of SNPs). Must be between 0 and 1.
`bias`	A vector of initial bias parameters. Should be the same length as the number of columns of `refmat` (number of SNPs). Must be greater than 0.
`od`	A vector of initial overdispersion parameters. Should be the same length as the number of columns of `refmat` (number of SNPs). Must be between 0 and 1.

Value

A three dimensional array. The rows index the individuals, the columns index the SNPs, and the third dimension indexes the genotypes. This is the log-likelihood for each individual/snp/genotype combination.

`compute_all_phifk` *Computes*

$$\Phi\hat{}-1(F(k|K, \alpha\_j, \rho\_i))$$

for all possible (i,j,k).

Description

Computes

$$\Phi^{-1}(F(k|K, \alpha_j, \rho_i))$$

for all possible (i,j,k).

Usage

```
compute_all_phifk(alpha, rho, ploidy)
```

Arguments

`alpha`	A vector whose jth element is the allele frequency of SNP j.
`rho`	A vector whose ith element is the inbreeding coefficient of individual i.
`ploidy`	The ploidy of the species.

Value

A three dimensional array. The rows index the individuals, the columns index the SNPs, and the third dimension indexes the genotypes. Computes the "continuous genotype".

Author(s)

David Gerard

`compute_all_post_prob` *Computes every posterior probability for each dosage level for each individual at each SNP.*

Description

Computes every posterior probability for each dosage level for each individual at each SNP.

Usage

```
compute_all_post_prob(ploidy, mu, sigma2, alpha, rho)
```

Arguments

`ploidy`	The ploidy of the species.
`mu`	A matrix of variational posterior means. The rows index the individuals and the columns index the SNPs.
`sigma2`	A matrix of variational posterior variances. The rows index the individuals and the columns index the SNPs.
`alpha`	A vector of allele frequencies for all SNPs.
`rho`	A vector of inbreeding coefficients for all individuals.

Value

An array. The rows index the individuals, the columns index the SNPS, and the third dimension indexes the genotypes. Element (i, j, k) is the return of `post_prob`.

Author(s)

David Gerard

`convolve` *Convolution between two discrete probability mass functions with support on 0:K.*

Description

Convolution between two discrete probability mass functions with support on 0:K.

Usage

```
convolve(x, y)
```

Arguments

`x`	The first probability vector. The ith element is the probability of i - 1.
`y`	The second probability vector. The ith element is the probability of i - 1.

Value

A vector that is the convolution of `x` and `y`. The ith element is the probability of i - 1.

Author(s)

David Gerard

See Also

`convolve` for a more generic convolution function.

Examples

```
x <- c(1 / 6, 2 / 6, 3 / 6)
y <- c(1 / 9, 2 / 9, 6 / 9)
convolve(x, y)
stats::convolve(x, rev(y), type = "o")
```

`dbernbinom` *Special case of betabinomial where the beta is bernoulli mu.*

Description

Special case of betabinomial where the beta is bernoulli mu.

Usage

```
dbernbinom(x, size, mu, log)
```

Arguments

`x`	The quantile.
`size`	The total number of draws.
`mu`	The mean of the beta.
`log`	A logical. Should we return the log of the density `TRUE` or not `FALSE`?

Value

The density of the Bernoulli-binomial.

Author(s)

David Gerard

`dbetabinom` *The Beta-Binomial Distribution*

Description

Density, distribution function, quantile function and random generation for the beta-binomial distribution when parameterized by the mean `mu` and the overdispersion parameter `rho` rather than the typical shape parameters.

Usage

```
dbetabinom(x, size, mu, rho, log)

pbetabinom(q, size, mu, rho, log_p)

qbetabinom(p, size, mu, rho)

rbetabinom(n, size, mu, rho)
```

Arguments

`x, q`	A vector of quantiles.
`size`	A vector of sizes.
`mu`	Either a scalar of the mean for each observation, or a vector of means of each observation, and thus the same length as `x` and `size`. This must be between 0 and 1.
`rho`	Either a scalar of the overdispersion parameter for each observation, or a vector of overdispersion parameters of each observation, and thus the same length as `x` and `size`. This must be between 0 and 1.
`log, log_p`	A logical vector either of length 1 or the same length as `x` and `size`. This determines whether to return the log probabilities for all observations (in the case that its length is 1) or for each observation (in the case that its length is that of `x` and `size`).
`p`	A vector of probabilities.
`n`	The number of observations.

Details

Let μ and ρ be the mean and overdispersion parameters. Let α and β be the usual shape parameters of a beta distribution. Then we have the relation

$$\mu = \alpha/(\alpha + \beta),$$

and

$$\rho = 1/(1 + \alpha + \beta).$$

This necessarily means that

$$\alpha = \mu(1 - \rho)/\rho,$$

and

$$\beta = (1 - \mu)(1 - \rho)/\rho.$$

Value

Either a random sample (`rbetabinom`), the density (`dbetabinom`), the tail probability (`pbetabinom`), or the quantile (`qbetabinom`) of the beta-binomial distribution.

Functions

- `dbetabinom`: Density function.
- `pbetabinom`: Distribution function.
- `qbetabinom`: Quantile function.
- `rbetabinom`: Random generation.

Author(s)

David Gerard

Examples

```
x <- rbetabinom(n = 10, size = 10, mu = 0.1, rho = 0.01)
dbetabinom(x = 1, size = 10, mu = 0.1, rho = 0.01, log = FALSE)
pbetabinom(q = 1, size = 10, mu = 0.1, rho = 0.01, log_p = FALSE)
qbetabinom(p = 0.6, size = 10, mu = 0.1, rho = 0.01)
```

---

dbetabinom_alpha_beta_double

Density function of betabinomial with the shape parameterizations

---

Description

Density function of betabinomial with the shape parameterizations

Usage

```
dbetabinom_alpha_beta_double(x, size, alpha, beta, log)
```

Arguments

x	The quantile.
size	The total number of draws.
alpha	The first shape parameter.
beta	The second shape parameter.
log	A logical. Should we return the log of the density TRUE or not FALSE?

Value

The density of the beta-binomial.

Author(s)

David Gerard

`dbetabinom_double` *The density function of the beta-binomial distribution.*

Description

The density function of the beta-binomial distribution.

Usage

```
dbetabinom_double(x, size, mu, rho, log)
```

Arguments

`x`	The quantile.
`size`	The total number of draws.
`mu`	The mean of the beta.
`rho`	The overdispersion parameter of the beta.
`log`	A logical. Should we return the log of the density `TRUE` or not `FALSE`?

Value

The density of the beta-binomial.

Author(s)

David Gerard

`dc_dtau` *Derivative of $c = (1 - \tau)/\tau$ with respect to τ.*

Description

Derivative of $c = (1 - \tau)/\tau$ with respect to τ.

Usage

```
dc_dtau(tau)
```

Arguments

`tau`	The overdispersion parameter.

Value

A double.

Author(s)

David Gerard

See Also

`dlbeta_dc`, `dlbeta_dtau`

`df_deps` *Derivative of f with respect to eps.*

Description

Derivative of f with respect to eps.

Usage

```
df_deps(p, eps)
```

Arguments

`p`	The allele dosage.
`eps`	The sequencing error rate.

Value

A double.

Author(s)

David Gerard

`dlbeta_dc` *Derivative of the log-beta density with respect to c where $c = (1-\tau)/\tau$ where τ is the overdispersion parameter.*

Description

Derivative of the log-beta density with respect to c where $c = (1 - \tau)/\tau$ where τ is the overdispersion parameter.

Usage

```
dlbeta_dc(x, n, xi, c)
```

Arguments

x	The number of successes observed
n	The total number of trials observed.
xi	The mean of the beta-binomial.
c	$(1-\tau)/\tau$ where τ is the overdispersion parameter.

Value

A double.

Author(s)

David Gerard

See Also

dbetabinom_double, dlbeta_dtau, dc_dtau.

---

dlbeta_deps — *Derivative of the log-beta-binomial density with respect to the sequencing error rate.*

Description

Derivative of the log-beta-binomial density with respect to the sequencing error rate.

Usage

```
dlbeta_deps(x, n, p, eps, h, tau)
```

Arguments

x	The number of successes.
n	The number of trials.
p	The allele dosage.
eps	The sequencing error rate
h	The bias parameter.
tau	The overdispersion parameter.

Value

A double.

Author(s)

David Gerard

dlbeta_dh *Derivative of log-betabinomial density with respect to bias parameter.*

Description

Derivative of log-betabinomial density with respect to bias parameter.

Usage

```
dlbeta_dh(x, n, p, eps, h, tau)
```

Arguments

x	The number of successes.
n	The number of trials.
p	The allele dosage.
eps	The sequencing error rate
h	The bias parameter.
tau	The overdispersion parameter.

Value

A double.

Author(s)

David Gerard

dlbeta_dtau *Derivative of the log-beta-binomial density with respect to the overdispersion parameter.*

Description

Derivative of the log-beta-binomial density with respect to the overdispersion parameter.

Usage

```
dlbeta_dtau(x, n, p, eps, h, tau)
```

Arguments

`x`	The number of successes.
`n`	The number of trials.
`p`	The allele dosage.
`eps`	The sequencing error rate
`h`	The bias parameter.
`tau`	The overdispersion parameter.

Value

A double.

Author(s)

David Gerard

See Also

`dlbeta_dc`, `dc_dtau`, `dbetabinom_double`.

`dlbeta_dxi` *Derivative of the log-betabinomial density with respect to the mean of the underlying beta.*

Description

Derivative of the log-betabinomial density with respect to the mean of the underlying beta.

Usage

```
dlbeta_dxi(x, n, xi, tau)
```

Arguments

`x`	The number of successes.
`n`	The number of trials.
`xi`	The mean of the underlying beta.
`tau`	The overdispersion parameter.

Value

A double.

Author(s)

David Gerard

doutdist	*The outlier distribution we use. Right now it is just a beta binomial with mean 1/2 and od 1/3 (so underlying beta is just a uniform from 0 to 1).*

Description

The outlier distribution we use. Right now it is just a beta binomial with mean 1/2 and od 1/3 (so underlying beta is just a uniform from 0 to 1).

Usage

```
doutdist(x, n, logp)
```

Arguments

x	The number of reference counts.
n	The total number of read-counts.
logp	Return the log density `TRUE` or not `FALSE`?

Value

A double. The outlier density value.

Author(s)

David Gerard

dpen_deps

dpen_deps *Derivative of*

$$-log(\epsilon(1-\epsilon)) - (logit(\epsilon) - \mu\_\epsilon)\hat{}2/(2\sigma\_\epsilon\hat{}2)$$

with respect to ϵ.

Description

Derivative of

$$-log(\epsilon(1-\epsilon)) - (logit(\epsilon) - \mu_\epsilon)^2/(2\sigma_\epsilon^2)$$

with respect to ϵ.

Usage

```
dpen_deps(eps, mu_eps, sigma2_eps)
```

Arguments

`eps`	The current sequencing error rate.
`mu_eps`	The mean of the logit of the sequencing error rate.
`sigma2_eps`	The variance of the logit of the sequencing error rate.

Value

A double.

Author(s)

David Gerard

See Also

`pen_seq_error` which this is a derivative for.

`dpen_dh` *Derivative of*

$$-log(h) - (log(h) - \mu\_h)\hat{}2/(2\sigma\_h\hat{}2)$$

with respect to h.

Description

Derivative of

$$-log(h) - (log(h) - \mu_h)^2/(2\sigma_h^2)$$

with respect to h.

Usage

```
dpen_dh(h, mu_h, sigma2_h)
```

Arguments

`h`	The current bias parameter.
`mu_h`	The mean of the log-bias.
`sigma2_h`	The variance of the log-bias.

Value

A double.

Author(s)

David Gerard

See Also

`pen_bias` which this is a derivative for.

`dr_pen` *Penalty used in `update_dr`.*

Description

A dirichlet prior on `pairweights`. Returns log density.

Usage

```
dr_pen(pairweights, mixing_pen)
```

Arguments

`pairweights`	The mixing proportions to penalize.
`mixing_pen`	The corresponding penalties.

Author(s)

David Gerard

See Also

`update_dr`

`dxi_df` *Derivative of xi with respect to f.*

Description

Derivative of xi with respect to f.

Usage

```
dxi_df(h, f)
```

Arguments

`h`	The bias parameter.
`f`	The post-sequencing error rate adjusted probability of an A.

Value

A double.

Author(s)

David Gerard

`dxi_dh` *Derivative of xi-function with respect to bias parameter.*

Description

Derivative of xi-function with respect to bias parameter.

Usage

```
dxi_dh(p, eps, h)
```

Arguments

`p`	The dosage (between 0 and 1).
`eps`	The sequencing error rate.
`h`	The bias parameter.

Value

A double.

Author(s)

David Gerard

`elbo` *The evidence lower bound*

Description

The evidence lower bound

Usage

```
elbo(warray, lbeta_array, cor_inv, postmean, postvar, bias, seq, mean_bias,
  var_bias, mean_seq, var_seq, ploidy)
```

Arguments

`warray`	An three-way array. The (i,j,k)th entry is the variational posterior probability that individual i at SNP j has dosage k - 1. See `compute_all_post_prob`.
`lbeta_array`	A three-way array. The (i,j,k)th entry is the log-density of the betabinomial for individual i at SNP j and dosage k - 1. See `compute_all_log_bb`.
`cor_inv`	The inverse of the correlation matrix.
`postmean`	A matrix. The (i,j)th entry is the variational posterior mean for individual i at SNP j.
`postvar`	A matrix. The (i,j)th entry is the variational posterior variance for individual i at SNP j.
`bias`	A vector. The jth entry is the allele bias for SNP j.
`seq`	A vector. The jth entry is the sequencing error rate at SNP j.
`mean_bias`	The prior mean on the log-bias.
`var_bias`	The prior variance on the log-bias.
`mean_seq`	The prior mean on the logit of the sequencing error rate.
`var_seq`	The prior variance on the logit of the sequencing error rate.
`ploidy`	The ploidy of the species.

Value

A double. The evidence lower-bound that `mupdog` maximizes.

Author(s)

David Gerard

`eta_double` *Adjusts allele dosage* `p` *by the sequencing error rate* `eps`.

Description

Adjusts allele dosage `p` by the sequencing error rate `eps`.

Usage

```
eta_double(p, eps)
```

Arguments

`p`	The allele dosage.
`eps`	The sequencing error rate.

Value

The probability of a reference reed adjusted by the sequencing error rate.

Author(s)

David Gerard

`eta_fun` *Adjusts allele dosage* `p` *by the sequencing error rate* `eps`.

Description

Adjusts allele dosage `p` by the sequencing error rate `eps`.

Usage

```
eta_fun(p, eps)
```

Arguments

`p`	A vector of allele dosages.
`eps`	The sequencing error rate. Must either be of length 1 or the same length as `p`.

Value

A vector of probabilities of a reference read adjusted by the sequencing error rate.

Author(s)

David Gerard

`expit` *The expit (logistic) function.*

Description

The expit (logistic) function.

Usage

```
expit(x)
```

Arguments

`x`	A double.

Value

The expit (logistic) of `x`.

Author(s)

David Gerard

`f1_obj` *Objective for mixture of known dist and uniform dist.*

Description

Objective for mixture of known dist and uniform dist.

Usage

```
f1_obj(alpha, pvec, weight_vec)
```

Arguments

`alpha`	The mixing weight.
`pvec`	The known distribtuion (e.g. from assuming an F1 population).
`weight_vec`	A vector of weights.

Value

The objective when updating `pivec` when `model = "f1"` or `model = "s1"` in `flexdog_full`.

Author(s)

David Gerard

`flexdog` *Flexible genotyping for polyploids from next-generation sequencing data.*

Description

Genotype polyploid individuals from next generation sequencing (NGS) data while assuming the genotype distribution is one of several forms. `flexdog` does this while accounting for allele bias, overdispersion, sequencing error, and possibly outlying observations (if `model = "f1"` or `model = "s1"`).

Usage

```
flexdog(refvec, sizevec, ploidy, model = c("norm", "hw", "bb", "ash", "s1",
  "s1pp", "f1", "f1pp", "flex", "uniform"), p1ref = NULL, p1size = NULL,
  p2ref = NULL, p2size = NULL, bias_init = exp(c(-1, -0.5, 0, 0.5, 1)),
  verbose = TRUE, outliers = FALSE, ...)
```

Arguments

`refvec`	A vector of counts of reads of the reference allele.
`sizevec`	A vector of total counts.
`ploidy`	The ploidy of the species. Assumed to be the same for each individual.
`model`	What form should the prior (genotype distribution) take? See Details for possible values.
`p1ref`	The reference counts for the first parent if `model = "f1"` (or `model = "f1pp"`), or for the only parent if `model = "s1"` (or `model = "s1pp"`).
`p1size`	The total counts for the first parent if `model = "f1"` (or `model = "f1pp"`), or for the only parent if `model = "s1"` (or `model = "s1pp"`).
`p2ref`	The reference counts for the second parent if `model = "f1"` (or `model = "f1pp"`).
`p2size`	The total counts for the second parent if `model = "f1"` (or `model = "f1pp"`).
`bias_init`	A vector of initial values for the bias parameter over the multiple runs of `flexdog_full`.
`verbose`	Should we output more (`TRUE`) or less (`FALSE`)?
`outliers`	A logical. Should we allow for the inclusion of outliers (`TRUE`) or not (`FALSE`). Only supported when `model = "f1"` or `model = "s1"`. I wouldn't recommend it for any other model anyway.
`...`	Additional parameters to pass to `flexdog_full`.

Details

Possible values of the genotype distribution (values of `model`) are:

`"norm"` A distribution whose genotype frequencies are proportional to the density value of a normal with some mean and some standard deviation. Unlike the `"bb"` and `"hw"` options, this will allow for distributions both more and less dispersed than a binomial. This seems to be the most robust to violations in modeling assumptions, and so is the default.

`"hw"` A binomial distribution that results from assuming that the population is in Hardy-Weinberg equilibrium (HWE). This actually does pretty well even when there are minor to moderate deviations from HWE.

`"bb"` A beta-binomial distribution. This is an overdispersed version of `"hw"` and can be derived from a special case of the Balding-Nichols model.

`"ash"` Any unimodal prior. This can sometimes be sensitive to violations in modeling assumptions, but tends to work better than the `"flex"` option.

`"s1"` This prior assumes the individuals are all full-siblings resulting from one generation of selfing. I.e. there is only one parent. This model assumes a particular type of meiotic behavior: polysomic inheritance with bivalent, non-preferential pairing. Since this is a pretty strong and well-founded prior, we allow `outliers = TRUE` when `model = "s1"`.

`"s1pp"` The same as `"s1"` but accounts for possible (and arbitrary levels of) preferential pairing during meiosis. The only supported values of `ploidy` right now for this option are `4` and `6`.

`"f1"` This prior assumes the individuals are all full-siblings resulting from one generation of a bi-parental cross. This model assumes a particular type of meiotic behavior: polysomic inheritance with bivalent, non-preferential pairing. Since this is a pretty strong and well-founded prior, we allow `outliers = TRUE` when `model = "f1"`.

`"f1pp"` The same as `"f1"` but accounts for possible (and arbitrary levels of) preferential pairing during meiosis. This option is mostly untested for values of `ploidy` greater than `6`.

`"flex"` Generically any categorical distribution. Theoretically, this works well if you have a lot of individuals. In practice, it seems to be less robust to violations in modeling assumptions.

`"uniform"` A discrete uniform distribution. This should never be used in practice.

You might think a good default is `model = "uniform"` because it is somehow an "uninformative prior." But it is very informative and tends to work horribly in practice. The intuition is that it will estimate the allele bias and sequencing error rates so that the estimated genotypes are approximately uniform (since we are assuming that they are approximately uniform). This will usually result in unintuitive genotyping since most populations don't have a uniform genotype distribution. I include it as an option only for completeness. Please don't use it.

The value of `prop_mis` is a very intuitive measure for the quality of the SNP. `prop_mis` is the posterior proportion of individuals mis-genotyped. So if you want only SNPS that accurately genotype, say, 95% of the individuals, you could discard all SNPs with a `prop_mis` over `0.05`.

The value of `maxpostprob` is a very intuitive measure for the quality of the genotype estimate of an individual. This is the posterior probability of correctly genotyping the individual when using `geno` (the posterior mode) as the genotype estimate. So if you want to correctly genotype, say, 95% of individuals, you could discard all individuals with a `maxpostprob` of under `0.95`. However, if you are just going to impute missing genotypes later, you might consider not discarding any individuals as `flexdog`'s genotype estimates will probably be more accurate than other more naive approaches, such as imputing using the grand mean.

In most datasets I've examined, allelic bias is a major issue. However, you may fit the model assuming no allelic bias by setting `update_bias = FALSE` and `bias_init = 1`.

Prior to using `flexdog`, during the read-mapping step, you could try to get rid of allelic bias by using WASP (`https://doi.org/10.1101/011221`). If you are successful in removing the allelic bias (because its only source was the read-mapping step), then setting `update_bias = FALSE` and `bias_init = 1` would be reasonable. You can visually inspect SNPs for bias by using `plot_geno`.

`flexdog`, like most methods, is invariant to which allele you label as the "reference" and which you label as the "alternative". That is, if you set `refvec` with the number of alternative read-counts, then the resulting genotype estimates will be the estimated allele dosage of the alternative allele.

Value

An object of class `flexdog`, which consists of a list with some or all of the following elements:

`bias` The estimated bias parameter.

`seq` The estimated sequencing error rate.

`od` The estimated overdispersion parameter.

`num_iter` The number of EM iterations ran. You should be wary if this equals `itermax`.

`llike` The maximum marginal log-likelihood.

`postmat` A matrix of posterior probabilities of each genotype for each individual. The rows index the individuals and the columns index the allele dosage.

`gene_dist` The estimated genotype distribution. The `i`th element is the proportion of individuals with genotype `i-1`. If `outliers = TRUE`, then this is conditional on the point not being an outlier.

`par` A list of the final estimates of the parameters of the genotype distribution. The elements included in `par` depends on the value of `model`:

- `model = "norm"`: `mu` is the normal mean and `sigma` is the normal standard deviation (not variance).
- `model = "hw"`: `alpha` is the major allele frequency.
- `model = "bb"`: `alpha` is the major allele frequency and `tau` is the overdisperion parameter (see the description of `rho` in the Details of `betabinom`).
- `model = "ash"`: `par` is an empty list.
- `model = "s1"`: `pgeno` is the allele dosage of the parent and `alpha` is the mixture proportion of the discrete uniform (included and fixed at a small value mostly for numerical stability reasons). See the description of `fs1_alpha` in `flexdog_full`.
- `model = "s1pp"`: `pgeno` is the allele dosage of the parent and `p1_pair_weights` contains a vector of mixing weights where element `i` is the mixing proportion for the segregation distribution in row `i` of `get_bivalent_probs(ploidy)$probmat[get_bivalent_probs(ploidy)$lvec == pgeno`
- `model = "f1"`: `p1geno` is the allele dosage of the first parent, `p2geno` is the allele dosage of the second parent, and `alpha` is the mixture proportion of the discrete uniform (included and fixed at a small value mostly for numerical stability reasons). See the description of `fs1_alpha` in `flexdog_full`.
- `model = "f1pp"`: `p1geno` is the allele dosage of the first parent, `p2geno` is the allele dosage of the second parent, `p1_pair_weights` contains a vector of mixing weights where element `i` is the mixing proportion for the segregation distribution for parent 1 in row `i` of `get_bivalent_probs(ploidy)$probmat[get_bivalent_probs(ploidy)$lvec == p1geno, , drop = FALSE` and `p2_pair_weights` contains a vector of mixing weights where element `i` is the mixing proportion for the segregation distribution for parent 2 in row `i` of `get_bivalent_probs(ploidy)$probmat[get_`
- `model = "flex"`: `par` is an empty list.
- `model = "uniform"`: `par` is an empty list.

`geno` The posterior mode genotype. These are your genotype estimates.

`maxpostprob` The maximum posterior probability. This is equivalent to the posterior probability of correctly genotyping each individual.

`postmean` The posterior mean genotype. In downstream association studies, you might want to consider using these estimates.

`input$refvec` The value of `refvec` provided by the user.

`input$sizevec` The value of `sizevec` provided by the user.

`input$ploidy` The value of `ploidy` provided by the user.

`input$model` The value of `model` provided by the user.

`input$p1ref` The value of `p1ref` provided by the user.

`input$p1size` The value of `p1size` provided by the user.

`input$p2ref` The value of `p2ref` provided by the user.

`input$p2size` The value of `p2size` provided by the user.

`prop_mis` The posterior proportion of individuals genotyped incorrectly.

`out_prop` The estimated proportion of points that are outliers. Only available if `outliers = TRUE`.

`prob_out` The ith element is the posterior probability that individual i is an outlier. Only available if `outliers = TRUE`.

Author(s)

David Gerard

References

Gerard, David, Luis Felipe Ventorim Ferrao, Antonio Augusto Franco Garcia, and Matthew Stephens. 2018. "Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids from Messy Sequencing Data." *bioRxiv*. Cold Spring Harbor Laboratory. doi:10.1101/281550.

See Also

Run `browseVignettes(package = "updog")` in R for example usage. Other useful functions include:

`flexdog_full` For additional parameter options when running `flexdog`.

`rgeno` For simulating genotypes under the allowable prior models in `flexdog`.

`rflexdog` For simulating read-counts under the assumed likelihood model in `flexdog`.

`plot.flexdog` For plotting the output of `flexdog`.

`oracle_mis` For calculating the oracle genotyping error rates. This is useful for read-depth calculations *before* collecting data. After you have data, using the value of `prop_mis` is better.

`oracle_cor` For calculating the correlation between the true genotypes and an oracle estimator (useful for read-depth calculations *before* collecting data).

Examples

```
## An S1 population where the first individual
## is the parent. Fit assuming outliers.
data("snpdat")
ploidy  <- 6
refvec  <- snpdat$counts[snpdat$snp == "SNP3"]
sizevec <- snpdat$size[snpdat$snp == "SNP3"]
fout    <- flexdog(refvec   = refvec[-1],
                   sizevec  = sizevec[-1],
                   ploidy   = ploidy,
                   model    = "s1",
                   p1ref    = refvec[1],
                   p1size   = sizevec[1],
                   outliers = TRUE)
```

```
plot(fout)



## A natural population. We will assume a
## normal prior since there are so few
## individuals.
data("uitdewilligen")
ploidy  <- 4
refvec  <- uitdewilligen$refmat[, 1]
sizevec <- uitdewilligen$sizemat[, 1]
fout    <- flexdog(refvec  = refvec,
                   sizevec = sizevec,
                   ploidy  = ploidy,
                   model   = "norm")
plot(fout)
```

`flexdog_full` *Flexible genotyping for polyploids from next-generation sequencing data.*

Description

Genotype polyploid individuals from next generation sequencing (NGS) data while assuming the genotype distribution is one of several forms. `flexdog` does this while accounting for allele bias, overdispersion, sequencing error, and possibly outlying observations (if `model = "f1"` or `model = "s1"`). This function has more options than `flexdog` and is only meant for expert users.

Usage

```
flexdog_full(refvec, sizevec, ploidy, model = c("norm", "hw", "bb", "ash",
  "s1", "s1pp", "f1", "f1pp", "flex", "uniform"), verbose = TRUE,
  mean_bias = 0, var_bias = 0.7^2, mean_seq = -4.7, var_seq = 1,
  seq = 0.005, bias = 1, od = 0.001, update_bias = TRUE,
  update_seq = TRUE, update_od = TRUE, mode = NULL, use_cvxr = FALSE,
  itermax = 200, tol = 10^-4, fs1_alpha = 10^-3, ashpen = 10^-6,
  p1ref = NULL, p1size = NULL, p2ref = NULL, p2size = NULL,
  outliers = FALSE)
```

Arguments

`refvec`	A vector of counts of reads of the reference allele.
`sizevec`	A vector of total counts.
`ploidy`	The ploidy of the species. Assumed to be the same for each individual.

`model`	What form should the prior (genotype distribution) take? See Details for possible values.
`verbose`	Should we output more (`TRUE`) or less (`FALSE`)?
`mean_bias`	The prior mean of the log-bias.
`var_bias`	The prior variance of the log-bias.
`mean_seq`	The prior mean of the logit of the sequencing error rate.
`var_seq`	The prior variance of the logit of the sequencing error rate.
`seq`	The starting value of the sequencing error rate.
`bias`	The starting value of the bias.
`od`	The starting value of the overdispersion parameter.
`update_bias`	A logical. Should we update `bias` (`TRUE`), or not (`FALSE`)?
`update_seq`	A logical. Should we update `seq` (`TRUE`), or not (`FALSE`)?
`update_od`	A logical. Should we update `od` (`TRUE`), or not (`FALSE`)?
`mode`	The mode if `model = "ash"`. If not provided, `flexdog` will estimate the mode. This is the starting point of the allele frequency if `model = "hw"`. This should be `NULL` for all other options of `model`.
`use_cvxr`	A logical. If `model = "ash"`, then do you want to use the EM algorithm (`FALSE`) or a convex optimization program using the package CVXR (`TRUE`)? Only available if CVXR is installed. Setting `use_cvxr` to `TRUE` is generally slower than setting it to `FALSE`.
`itermax`	The maximum number of EM iterations to run for each mode (if `model = "ash"`) or the total number of EM iterations to run (for any other value of `model`).
`tol`	The tolerance stopping criterion. The EM algorithm will stop if the difference in the log-likelihoods between two consecutive iterations is less than `tol`.
`fs1_alpha`	The value at which to fix the mixing proportion for the uniform component when `model = "f1"`, `model = "f1pp"`, `model = "s1"`, or `model = "s1pp"`. I would recommend some small value such as `10^-3`.
`ashpen`	The penalty to put on the unimodal prior. Larger values shrink the unimodal prior towards the discrete uniform distribution.
`p1ref`	The reference counts for the first parent if `model = "f1"` (or `model = "f1pp"`), or for the only parent if `model = "s1"` (or `model = "s1pp"`).
`p1size`	The total counts for the first parent if `model = "f1"` (or `model = "f1pp"`), or for the only parent if `model = "s1"` (or `model = "s1pp"`).
`p2ref`	The reference counts for the second parent if `model = "f1"` (or `model = "f1pp"`).
`p2size`	The total counts for the second parent if `model = "f1"` (or `model = "f1pp"`).
`outliers`	A logical. Should we allow for the inclusion of outliers (`TRUE`) or not (`FALSE`). Only supported when `model = "f1"` or `model = "s1"`. I wouldn't recommend it for any other model anyway.

Details

Possible values of the genotype distribution (values of `model`) are:

`"norm"` A distribution whose genotype frequencies are proportional to the density value of a normal with some mean and some standard deviation. Unlike the `"bb"` and `"hw"` options, this will allow for distributions both more and less dispersed than a binomial. This seems to be the most robust to violations in modeling assumptions, and so is the default.

`"hw"` A binomial distribution that results from assuming that the population is in Hardy-Weinberg equilibrium (HWE). This actually does pretty well even when there are minor to moderate deviations from HWE.

`"bb"` A beta-binomial distribution. This is an overdispersed version of `"hw"` and can be derived from a special case of the Balding-Nichols model.

`"ash"` Any unimodal prior. This can sometimes be sensitive to violations in modeling assumptions, but tends to work better than the `"flex"` option.

`"s1"` This prior assumes the individuals are all full-siblings resulting from one generation of selfing. I.e. there is only one parent. This model assumes a particular type of meiotic behavior: polysomic inheritance with bivalent, non-preferential pairing. Since this is a pretty strong and well-founded prior, we allow `outliers = TRUE` when `model = "s1"`.

`"s1pp"` The same as `"s1"` but accounts for possible (and arbitrary levels of) preferential pairing during meiosis. The only supported values of `ploidy` right now for this option are `4` and `6`.

`"f1"` This prior assumes the individuals are all full-siblings resulting from one generation of a bi-parental cross. This model assumes a particular type of meiotic behavior: polysomic inheritance with bivalent, non-preferential pairing. Since this is a pretty strong and well-founded prior, we allow `outliers = TRUE` when `model = "f1"`.

`"f1pp"` The same as `"f1"` but accounts for possible (and arbitrary levels of) preferential pairing during meiosis. This option is mostly untested for values of `ploidy` greater than `6`.

`"flex"` Generically any categorical distribution. Theoretically, this works well if you have a lot of individuals. In practice, it seems to be less robust to violations in modeling assumptions.

`"uniform"` A discrete uniform distribution. This should never be used in practice.

You might think a good default is `model = "uniform"` because it is somehow an "uninformative prior." But it is very informative and tends to work horribly in practice. The intuition is that it will estimate the allele bias and sequencing error rates so that the estimated genotypes are approximately uniform (since we are assuming that they are approximately uniform). This will usually result in unintuitive genotyping since most populations don't have a uniform genotype distribution. I include it as an option only for completeness. Please don't use it.

The value of `prop_mis` is a very intuitive measure for the quality of the SNP. `prop_mis` is the posterior proportion of individuals mis-genotyped. So if you want only SNPS that accurately genotype, say, 95% of the individuals, you could discard all SNPs with a `prop_mis` over `0.05`.

The value of `maxpostprob` is a very intuitive measure for the quality of the genotype estimate of an individual. This is the posterior probability of correctly genotyping the individual when using `geno` (the posterior mode) as the genotype estimate. So if you want to correctly genotype, say, 95% of individuals, you could discard all individuals with a `maxpostprob` of under `0.95`. However, if you are just going to impute missing genotypes later, you might consider not discarding any individuals as `flexdog`'s genotype estimates will probably be more accurate than other more naive approaches, such as imputing using the grand mean.

In most datasets I've examined, allelic bias is a major issue. However, you may fit the model assuming no allelic bias by setting `update_bias = FALSE` and `bias_init = 1`.

Prior to using `flexdog`, during the read-mapping step, you could try to get rid of allelic bias by using WASP (`https://doi.org/10.1101/011221`). If you are successful in removing the allelic bias (because its only source was the read-mapping step), then setting `update_bias = FALSE` and `bias_init = 1` would be reasonable. You can visually inspect SNPs for bias by using `plot_geno`.

`flexdog`, like most methods, is invariant to which allele you label as the "reference" and which you label as the "alternative". That is, if you set `refvec` with the number of alternative read-counts, then the resulting genotype estimates will be the estimated allele dosage of the alternative allele.

Value

An object of class `flexdog`, which consists of a list with some or all of the following elements:

`bias` The estimated bias parameter.

`seq` The estimated sequencing error rate.

`od` The estimated overdispersion parameter.

`num_iter` The number of EM iterations ran. You should be wary if this equals `itermax`.

`llike` The maximum marginal log-likelihood.

`postmat` A matrix of posterior probabilities of each genotype for each individual. The rows index the individuals and the columns index the allele dosage.

`gene_dist` The estimated genotype distribution. The `i`th element is the proportion of individuals with genotype `i-1`. If `outliers = TRUE`, then this is conditional on the point not being an outlier.

`par` A list of the final estimates of the parameters of the genotype distribution. The elements included in `par` depends on the value of `model`:

- `model = "norm"`: `mu` is the normal mean and `sigma` is the normal standard deviation (not variance).
- `model = "hw"`: `alpha` is the major allele frequency.
- `model = "bb"`: `alpha` is the major allele frequency and `tau` is the overdisperion parameter (see the description of `rho` in the Details of `betabinom`).
- `model = "ash"`: `par` is an empty list.
- `model = "s1"`: `pgeno` is the allele dosage of the parent and `alpha` is the mixture proportion of the discrete uniform (included and fixed at a small value mostly for numerical stability reasons). See the description of `fs1_alpha` in `flexdog_full`.
- `model = "s1pp"`: `pgeno` is the allele dosage of the parent and `p1_pair_weights` contains a vector of mixing weights where element `i` is the mixing proportion for the segregation distribution in row `i` of `get_bivalent_probs(ploidy)$probmat[get_bivalent_probs(ploidy)$lvec == pgeno`
- `model = "f1"`: `p1geno` is the allele dosage of the first parent, `p2geno` is the allele dosage of the second parent, and `alpha` is the mixture proportion of the discrete uniform (included and fixed at a small value mostly for numerical stability reasons). See the description of `fs1_alpha` in `flexdog_full`.
- `model = "f1pp"`: `p1geno` is the allele dosage of the first parent, `p2geno` is the allele dosage of the second parent, `p1_pair_weights` contains a vector of mixing weights where element `i` is the mixing proportion for the segregation distribution for parent 1 in row `i` of

get_bivalent_probs(ploidy)$probmat[get_bivalent_probs(ploidy)$lvec == p1geno, , drop = FALSE
and p2_pair_weights contains a vector of mixing weights where element i is the mixing proportion for the segregation distribution for parent 2 in row i of get_bivalent_probs(ploidy)$probmat[get_

model = "flex": par is an empty list.

model = "uniform": par is an empty list.

geno The posterior mode genotype. These are your genotype estimates.

maxpostprob The maximum posterior probability. This is equivalent to the posterior probability of correctly genotyping each individual.

postmean The posterior mean genotype. In downstream association studies, you might want to consider using these estimates.

input$refvec The value of refvec provided by the user.

input$sizevec The value of sizevec provided by the user.

input$ploidy The value of ploidy provided by the user.

input$model The value of model provided by the user.

input$p1ref The value of p1ref provided by the user.

input$p1size The value of p1size provided by the user.

input$p2ref The value of p2ref provided by the user.

input$p2size The value of p2size provided by the user.

prop_mis The posterior proportion of individuals genotyped incorrectly.

out_prop The estimated proportion of points that are outliers. Only available if outliers = TRUE.

prob_out The ith element is the posterior probability that individual i is an outlier. Only available if outliers = TRUE.

Author(s)

David Gerard

References

Gerard, David, Luis Felipe Ventorim Ferrao, Antonio Augusto Franco Garcia, and Matthew Stephens. 2018. "Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids from Messy Sequencing Data." *bioRxiv*. Cold Spring Harbor Laboratory. doi:10.1101/281550.

See Also

Run browseVignettes(package = "updog") in R for example usage. Other useful functions include:

flexdog For a more user-friendly version of flexdog_full.

rgeno For simulating genotypes under the allowable prior models in flexdog.

rflexdog For simulating read-counts under the assumed likelihood model in flexdog.

plot.flexdog For plotting the output of flexdog.

oracle_mis For calculating the oracle genotyping error rates. This is useful for read-depth calculations *before* collecting data. After you have data, using the value of prop_mis is better.

oracle_cor For calculating the correlation between the true genotypes and an oracle estimator (useful for read-depth calculations *before* collecting data).

Examples

```
## A natural population. We will assume a
## normal prior since there are so few
## individuals.
data("uitdewilligen")
ploidy  <- 4
refvec  <- uitdewilligen$refmat[, 1]
sizevec <- uitdewilligen$sizemat[, 1]
fout    <- flexdog_full(refvec  = refvec,
                        sizevec = sizevec,
                        ploidy  = ploidy,
                        model   = "norm")
plot(fout)
```

`flexdog_obj` *Log-likelihood that `flexdog` maximizes.*

Description

Log-likelihood that `flexdog` maximizes.

Usage

```
flexdog_obj(probk_vec, refvec, sizevec, ploidy, seq, bias, od, mean_bias,
  var_bias, mean_seq, var_seq)
```

Arguments

`probk_vec`	The kth element is the prior probability of genotype k (when starting to count from 0).
`refvec`	A vector of counts of reads of the reference allele.
`sizevec`	A vector of total counts.
`ploidy`	The ploidy of the species. Assumed to be the same for each individual.
`seq`	The starting value of the sequencing error rate.
`bias`	The starting value of the bias.
`od`	The starting value of the overdispersion parameter.
`mean_bias`	The prior mean of the log-bias.
`var_bias`	The prior variance of the log-bias.
`mean_seq`	The prior mean of the logit of the sequencing error rate.
`var_seq`	The prior variance of the logit of the sequencing error rate.

Value

The objective (marginal log-likelihood) used in `flexdog_full`.

Author(s)

David Gerard

`flexdog_obj_out` *Log-likelihood that `flexdog` maximizes when outliers are present.*

Description

Log-likelihood that `flexdog` maximizes when outliers are present.

Usage

```
flexdog_obj_out(probk_vec, out_prop, refvec, sizevec, ploidy, seq, bias, od,
  mean_bias, var_bias, mean_seq, var_seq)
```

Arguments

`probk_vec`	The kth element is the prior probability of genotype k (when starting to count from 0).
`out_prop`	The probability of being an outlier.
`refvec`	A vector of counts of reads of the reference allele.
`sizevec`	A vector of total counts.
`ploidy`	The ploidy of the species. Assumed to be the same for each individual.
`seq`	The starting value of the sequencing error rate.
`bias`	The starting value of the bias.
`od`	The starting value of the overdispersion parameter.
`mean_bias`	The prior mean of the log-bias.
`var_bias`	The prior variance of the log-bias.
`mean_seq`	The prior mean of the logit of the sequencing error rate.
`var_seq`	The prior variance of the logit of the sequencing error rate.

Value

A double. The `flexdog` objective when `outliers = TRUE`.

Author(s)

David Gerard

See Also

`flexdog_obj` for the objective function without outliers.

flex_update_pivec *Update the distribution of genotypes from various models.*

Description

Update the distribution of genotypes from various models.

Usage

```
flex_update_pivec(weight_vec, model = c("hw", "bb", "norm", "ash", "f1", "s1",
  "f1pp", "s1pp", "f1ppdr", "s1ppdr", "flex", "uniform"), control)
```

Arguments

`weight_vec`	`colSums(wik_mat)` from `flexdog`. This is the sum of current posterior probabilities of each individual having genotype k.
`model`	What model are we assuming? See the description in `flexdog` for details.
`control`	A list of anything else needed to be passed. E.g. if `model = "ash"`, then `inner_weights` needs to be passed through `control` (see `get_inner_weights` for how to get this matrix).

Value

A list with the following elements

`pivec` The estimate of the genotype distribution.

`par` A list of estimated parameters. An empty list if the model does not contain any parameters other than `pivec`.

Author(s)

David Gerard

get_bivalent_probs *Returns segregation probabilities, pairing representation and number of ref alleles given the ploidy.*

Description

Returns segregation probabilities, pairing representation and number of ref alleles given the ploidy.

Usage

```
get_bivalent_probs(ploidy)
```

Arguments

`ploidy`	The ploidy of the individual. Should be even and greater than 0.

Value

A list of three elements

`probmat` The rows index the pairing configuration and the columns index the number of reference alleles segregating. The elements are the probability of segregating the given number of reference alleles in a given category.

`pmat` The pairing representation of the configuration.

`lvec` The number of reference alleles an individual has given their pairing configuration in `pmat`.

Author(s)

David Gerard

Examples

```
get_bivalent_probs(4)
```

`get_bivalent_probs_dr` *Double reduction version of* `get_bivalent_probs`.

Description

Double reduction version of `get_bivalent_probs`.

Usage

```
get_bivalent_probs_dr(ploidy)
```

Arguments

`ploidy`	The ploidy of the individual. Should be even and greater than 0.

Value

A list. The same elements as in `get_bivalent_probs`, augmented to include more scenarios, but with the additional element `penvec`. This is a logical vector that is `TRUE` if the corresponding rows of `probmat` and `pmat` and elements of `lvec` would be included in the non-double-reduction-model and is `FALSE` otherwise.

Author(s)

David Gerard

See Also

get_bivalent_probs.

---

get_conv_inner_weights
Get the inner weights used for the em update in update_pp_f1 *when there are more than two bivalent components for one of the parents.*

Description

Get the inner weights used for the em update in update_pp_f1 when there are more than two bivalent components for one of the parents.

Usage

```
get_conv_inner_weights(psegprob, psegmat)
```

Arguments

psegprob	One of the parents segregation probability vector.
psegmat	The other parent's segregation matrix.

Value

A matrix. The columns index the K components (aka individuals in the context of the local problem) and the rows index the bivalent components.

Author(s)

David Gerard

See Also

update_pp_f1 for where this is used. uni_em_const for where the weights are used (equivalent to lmat there).

get_dimname — *Returns a vector character strings that are all of the possible combinations of the reference allele and the non-reference allele.*

Description

Returns a vector character strings that are all of the possible combinations of the reference allele and the non-reference allele.

Usage

```
get_dimname(ploidy)
```

Arguments

`ploidy`	The ploidy of the species.

Value

For example, if `ploidy = 3` then this will return c("aaa", "Aaa", "AAa", "AAA")

Author(s)

David Gerard

get_hyper_weights — *Return mixture weights needed to obtain a hypergeometric distribution.*

Description

Obtains the mixing weights for the mixing distributions of `get_bivalent_probs` to return a hypergeometric distribution where `ploidy` is the population size, `ell` is the number of success states in the population, and `ploidy / 2` is the number of draws. If these are the mixing weights in the population, then there is no preferential pairing.

Usage

```
get_hyper_weights(ploidy, ell)
```

Arguments

`ploidy`	The ploidy of the individual.
`ell`	The number of reference alleles in the individual.

Value

A list with the following two elements:

`pmat` Reach row is a category and the columns index either aa, Aa, or AA.

`weightvec` The mixing weights for each row of pmat.

Author(s)

David Gerard

Examples

```
get_hyper_weights(4, 2)
```

`get_inner_weights` *Compute inner weights for updating the mixing proportions when using ash model.*

Description

The (i,k)th element is $1(k \in F(a,i))/|F(a,i)|$.

Usage

```
get_inner_weights(ploidy, mode)
```

Arguments

`ploidy`	The ploidy of the species. Assumed to be the same for each individual.
`mode`	The mode if `model = "ash"`. If not provided, `flexdog` will estimate the mode. This is the starting point of the allele frequency if `model = "hw"`. This should be `NULL` for all other options of `model`.

Value

A matrix of numerics. The weights used for the weighted EM algorithm in `flexdog_full`.

Author(s)

David Gerard

`get_probk_vec` *Obtain the genotype distribution given the distribution of discrete uniforms.*

Description

Obtain the genotype distribution given the distribution of discrete uniforms.

Usage

```
get_probk_vec(pivec, model, mode)
```

Arguments

`pivec`	The mixing probability of the i'th discrete uniform distribution.
`model`	What form should the prior (genotype distribution) take? See Details for possible values.
`mode`	The mode if `model = "ash"`. If not provided, `flexdog` will estimate the mode. This is the starting point of the allele frequency if `model = "hw"`. This should be `NULL` for all other options of `model`.

Value

A vector of numerics. Element k is the probability of genotype k.

Author(s)

David Gerard

See Also

`flexdog` where this is used.

`get_q_array` *Return the probabilities of an offspring's genotype given its parental genotypes for all possible combinations of parental and offspring genotypes. This is for species with polysomal inheritance and bivalent, non-preferential pairing.*

Description

Return the probabilities of an offspring's genotype given its parental genotypes for all possible combinations of parental and offspring genotypes. This is for species with polysomal inheritance and bivalent, non-preferential pairing.

Usage

```
get_q_array(ploidy)
```

Arguments

`ploidy`	A positive integer. The ploidy of the species.

Value

An three-way array of proportions. The (i, j, k)th element is the probability of an offspring having k - 1 reference alleles given that parent 1 has i - 1 reference alleles and parent 2 has j - 1 reference alleles. Each dimension of the array is `ploidy + 1`. In the dimension names, "A" stands for the reference allele and "a" stands for the alternative allele.

Author(s)

David Gerard

Examples

```
qarray <- get_q_array(6)
apply(qarray, c(1, 2), sum) ## should all be 1's.
```

`get_uni_rep` *Get the representation of a discrete unimodal probability distribution.*

Description

NB: In `get_uni_rep`, we count the mode starting at 1. In `get_probk_vec`, we count the mode starting at 0.

Usage

```
get_uni_rep(probvec)
```

Arguments

`probvec`	A probability vector. It assumes the probabilities are ordered according to the ordering of the discrete set.

Value

A list with the following elements

`pivec` The mixing weights for the unimodal representation.

`mode` The central value of the unimodal distribution.

Author(s)

David Gerard

See Also

`get_probk_vec` with option `model = "ash"` for the inverse of this function.

`get_wik_mat` *E-step in `flexdog`.*

Description

E-step in `flexdog`.

Usage

```
get_wik_mat(probk_vec, refvec, sizevec, ploidy, seq, bias, od)
```

Arguments

`probk_vec`	The vector of current prior probabilities of each genotype.
`refvec`	A vector of counts of reads of the reference allele.
`sizevec`	A vector of total counts.
`ploidy`	The ploidy of the species. Assumed to be the same for each individual.
`seq`	The starting value of the sequencing error rate.
`bias`	The starting value of the bias.
`od`	The starting value of the overdispersion parameter.

Value

A matrix of numerics. The rows index the individuals and the columns index the genotype. These weights are used in the EM algorithm (and is indeed the E-step) in `flexdog_full`.

Author(s)

David Gerard

See Also

`flexdog` for the full EM algorithm.

get_wik_mat_out — *E-step in flexdog where we now allow an outlier distribution.*

Description

E-step in flexdog where we now allow an outlier distribution.

Usage

```
get_wik_mat_out(probk_vec, out_prop, refvec, sizevec, ploidy, seq, bias, od)
```

Arguments

probk_vec	The vector of current prior probabilities of each genotype.
out_prop	The probability of being an outlier.
refvec	A vector of counts of reads of the reference allele.
sizevec	A vector of total counts.
ploidy	The ploidy of the species. Assumed to be the same for each individual.
seq	The starting value of the sequencing error rate.
bias	The starting value of the bias.
od	The starting value of the overdispersion parameter.

Value

Same as get_wik_mat but the last column is for the outlier class.

Author(s)

David Gerard

See Also

flexdog for the full EM algorithm. get_wik_mat for the equivalent function without outliers. doutdist for the outlier density function.

grad_for_eps *Gradient for* obj_for_eps.

Description

Gradient for obj_for_eps.

Usage

```
grad_for_eps(parvec, refvec, sizevec, ploidy, mean_bias, var_bias, mean_seq,
  var_seq, wmat, update_bias = TRUE, update_seq = TRUE, update_od = TRUE)
```

Arguments

parvec	A vector of length three. The first element is the sequencing error rate, the second element is the allele bias, and the third element is the overdispersion parameter.
refvec	A vector. The ith element is the reference count for the ith individual in the SNP.
sizevec	A vector. the ith element is the size count for the ith individual in the SNP/
ploidy	The ploidy of the species.
mean_bias	The prior mean of the log-bias.
var_bias	The prior variance of the log-bias
mean_seq	The prior mean of the logit sequencing error rate.
var_seq	The prior variance of the logit sequencing error rate.
wmat	The matrix of (variational) posterior probabilities for each dosage. The rows index the individuals and the columns index the dosage levels.
update_bias	A logical. This is not used in obj_for_eps, but sets the second element to 0.0 in grad_for_eps.
update_seq	A logical. This is not used in obj_for_eps, but sets the first element to 0.0 in grad_for_eps.
update_od	A logical. This is not used in obj_for_eps, but sets the third element to 0.0 in grad_for_eps.

Value

A double.

Author(s)

David Gerard

grad_for_mu_sigma2 — *Gradient for* obj_for_mu_sigma2 *with respect for* mu *and* sigma2.

Description

Gradient for obj_for_mu_sigma2 with respect for mu and sigma2.

Usage

```
grad_for_mu_sigma2(mu, sigma2, phifk_mat, cor_inv, log_bb_dense)
```

Arguments

mu	A vector, the ith element is the variational posterior mean of individual i for the SNP.
sigma2	A vector, the ith element is the variational posterior variance of individual i for the SNP.
phifk_mat	A matrix that contains the standard normal quantile of the beta-binomial cdf at dosage k for individual i. The rows index the individuals and the columns index the dosages.
cor_inv	The inverse of the underlying correlation matrix.
log_bb_dense	A matrix of log-densities of the beta binomial. The rows index the individuals and the columns index the allele dosage. Allele dosage goes from -1 to ploidy, so there are ploidy + 2 elements.

Value

A vector of length 2 * nind of numerics. The first element n elements are the partial derivatives with respect to mu and the second n elements are the partial derivatives with respect to sigma2 in obj_for_mu_sigma2.

Author(s)

David Gerard

grad_for_mu_sigma2_wrapper — *Gradient for* obj_for_mu_sigma2_wrapper *with respect for* muSigma2 *and a wrapper for* grad_for_mu_sigma2

Description

Gradient for obj_for_mu_sigma2_wrapper with respect for muSigma2 and a wrapper for grad_for_mu_sigma2

Usage

```
grad_for_mu_sigma2_wrapper(muSigma2, phifk_mat, cor_inv, log_bb_dense)
```

Arguments

`muSigma2`	A vector. The first half are mu and the second half are sigma2.
`phifk_mat`	A matrix that contains the standard normal quantile of the beta-binomial cdf at dosage k for individual i. The rows index the individuals and the columns index the dosages.
`cor_inv`	The inverse of the underlying correlation matrix.
`log_bb_dense`	A matrix of log-densities of the beta binomial. The rows index the individuals and the columns index the allele dosage. Allele dosage goes from -1 to ploidy, so there are ploidy + 2 elements.

Value

A vector of length 2 * nind of numerics. The first element n elements are the partial derivatives with respect to `mu` and the second n elements are the partial derivatives with respect to `sigma2` in `obj_for_mu_sigma2`.

Author(s)

David Gerard

`grad_for_weighted_lbb` *Gradient for* `obj_for_weighted_lbb`.

Description

Gradient for `obj_for_weighted_lbb`.

Usage

```
grad_for_weighted_lbb(parvec, ploidy, weight_vec)
```

Arguments

`parvec`	A vector of length 2. The first term is the current mean of the underlying beta. The second term is the current overdispersion parameter.
`ploidy`	The ploidy of the species.
`weight_vec`	A vector of length `ploidy + 1` that contains the weights for each component beta-binomial.

Value

A vector of length 2. The first component is the gradient of the mean of the underlying beta. The second component is the gradient of the overdispersion parameter of the underlying beta.

Author(s)

David Gerard

`grad_for_weighted_lnorm` *Gradient for* `obj_for_weighted_lnorm`*.*

Description

Gradient for `obj_for_weighted_lnorm`.

Usage

```
grad_for_weighted_lnorm(parvec, ploidy, weight_vec)
```

Arguments

`parvec`	A vector of length 2. The first term is the current mean of the underlying normal. The second term is the current standard deviation (not variance) of the normal.
`ploidy`	The ploidy of the species.
`weight_vec`	A vector of length `ploidy + 1` that contains the weights for each component beta-binomial.

Value

A vector of length 2. The first term is the derivative with respect to the mean, the second term is the derivative with respect to the standard deviation (not variance).

Author(s)

David Gerard

`initialize_pivec` *Initialize* `pivec` *for* `flexdog` *EM algorithm.*

Description

The key idea here is choosing the pi's so that the two modes have equal probability.

Usage

```
initialize_pivec(ploidy, mode, model = c("hw", "bb", "norm", "ash", "f1",
  "s1", "f1pp", "s1pp", "f1ppdr", "s1ppdr", "flex", "uniform"))
```

Arguments

`ploidy`	The ploidy of the species. Assumed to be the same for each individual.
`mode`	The mode if `model = "ash"`. If not provided, `flexdog` will estimate the mode. This is the starting point of the allele frequency if `model = "hw"`. This should be `NULL` for all other options of `model`.
`model`	What form should the prior (genotype distribution) take? See Details for possible values.

Value

A vector of numerics. The initial value of `pivec` used in `flexdog_full`.

Author(s)

David Gerard

See Also

`flexdog` for where this is used.

`is.flexdog` *Tests if an argument is a* `flexdog` *object.*

Description

Tests if an argument is a `flexdog` object.

Usage

```
is.flexdog(x)
```

Arguments

`x`	Anything.

Value

A logical. `TRUE` if `x` is a `flexdog` object, and `FALSE` otherwise.

Author(s)

David Gerard

Examples

```
is.flexdog("anything")
# FALSE
```

`is.mupdog` *Tests if its argument is a mupdog object.*

Description

Tests if its argument is a mupdog object.

Usage

```
is.mupdog(x)
```

Arguments

`x`	Anything.

Value

A logical. `TRUE` if `x` is a mupdog object, and `FALSE` otherwise.

Author(s)

David Gerard

Examples

```
is.mupdog("anything")
# FALSE
```

`logit` *The logit function.*

Description

The logit function.

Usage

```
logit(x)
```

Arguments

`x`	A double between 0 and 1.

Value

The logit of `x`.

Author(s)

David Gerard

`log_sum_exp` *Log-sum-exponential trick.*

Description

Log-sum-exponential trick.

Usage

```
log_sum_exp(x)
```

Arguments

`x`	A vector to log-sum-exp.

Value

The log of the sum of the exponential of the elements in `x`.

Author(s)

David Gerard

`log_sum_exp_2` *Log-sum-exponential trick using just two doubles.*

Description

Log-sum-exponential trick using just two doubles.

Usage

```
log_sum_exp_2(x, y)
```

Arguments

`x`	A double.
`y`	Another double.

Value

The log of the sum of the exponential of x and y.

Author(s)

David Gerard

mupdog *Multi-SNP updog.*

Description

A method to genotype autopolyploids using GBS or RAD-seq like data by accounting for correlations in the genotype distribution between the individuals.

Usage

```
mupdog(refmat, sizemat, ploidy, verbose = TRUE, mean_bias = 0,
  var_bias = 1, mean_seq = -4.7, var_seq = 1, seq = NULL, bias = NULL,
  od = NULL, allele_freq = NULL, inbreeding = NULL, cor_mat = NULL,
  postmean = NULL, postvar = NULL, update_cor = TRUE,
  update_inbreeding = TRUE, update_allele_freq = TRUE, num_core = 1,
  update_method = c("Brent", "L-BFGS-B"), control = list())
```

Arguments

`refmat`	A matrix of reference counts. The rows index the individuals and the columns index the SNPs.
`sizemat`	A matrix of total counts. The rows index the individuals and the columns index the SNPs. Should have the same dimensions as `refmat`.
`ploidy`	The ploidy of the species. To estimate the ploidy, re-run `mupdog` at various ploidy levels and choose the one with the largest ELBO. This assumes that the ploidy is the same for all individuals in the sample.
`verbose`	Should we print a lot of output `TRUE` or not `FALSE`?
`mean_bias`	The prior mean of the log-bias. Defaults to 0 (no bias).
`var_bias`	The prior variance on the log-bias. Defaults to 1. This roughly corresponds to likely bias values between 0.14 and 7.4. This is a far wider interval than what we observe in practice, thus making this prior rather uninformative. We usually observe bias values somewhere between 0.5 and 2.
`mean_seq`	The prior mean of the logit-sequencing-error-rate. Defaults to -4.7. This corresponds to a sequencing error rate of 0.009.
`var_seq`	The prior variance of the logit-sequencing-error-rate. Defaults to 1. This corresponds to likely values of 0.001 and 0.06. This upper bound is larger than what we would expect given the current state of next-gen-sequencing technology.
`seq`	A vector of initial sequencing errors. Should be the same length as the number of columns of `refmat` (number of SNPs). Must be between 0 and 1.
`bias`	A vector of initial bias parameters. Should be the same length as the number of columns of `refmat` (number of SNPs). Must be greater than 0.
`od`	A vector of initial overdispersion parameters. Should be the same length as the number of columns of `refmat` (number of SNPs). Must be between 0 and 1.

`allele_freq`	A vector of initial allele frequencies. Should be the same length as the number of columns of `refmat` (number of SNPs). Must be between 0 and 1.
`inbreeding`	A vector of initial individual-specific inbreeding coefficients. Should be the same length as the number of rows of `refmat` (number of individuals). Must be between 0 and 1.
`cor_mat`	Initial correlation matrix. Should have the same number of columns/rows as the number of individuals.
`postmean`	Initial variational posterior means. Should have the same dimensions as `refmat`.
`postvar`	Initial posterior variances. Should have the same dimensions as `refmat`.
`update_cor`	A logical. Should we update the underlying correlation matrix `TRUE` or not `FALSE`. It might be unwise to set this to `TRUE` if you have more individuals than SNPs.
`update_inbreeding`	A logical. Should we update the inbreeding coefficients `TRUE` or not `FALSE`?
`update_allele_freq`	A logical. Should we update the allele frequencies `TRUE` or not `FALSE`?
`num_core`	The number of cores to use if you want to run the optimization steps in parallel. If `num_core = 1`, then the optimization step will not be run in parallel.
`update_method`	What generic optimizer should we use to update `allele_freq` and `inbreeding`? Options are either `"Brent"` or `"L-BFGS-B"`. See `optim` for details on these optimizers.
`control`	A list of control parameters (`itermax`, `obj_tol`).

Details

Blischak et al (2017) developed a genotyping approach for autopolyploids that assumes a Balding-Nichols generative model (Balding and Nichols, 1997) on the genotypes. Using a different generative model, Gerard et al (2018) accounted for common issues in sequencing data ignored by previous researchers. Mupdog unites and extends these two approaches:

- Unite: We account for locus-specific allele-bias, locus-specific sequencing error, and locus-specific overdispersion while marginally assuming a Balding-Nichols generative model on the genotypes.
- Extend: We account for underlying correlations between the individuals using a Gaussian copula model.

Mupdog uses a variational Bayes approach to estimate all parameters of interest and the posterior probabilities of the genotypes for each individual at each locus.

Value

A list with some or all of the following elements:

`map_dosage` A matrix of numerics containing the variational maximum-a-posterior (MAP) genotypes (allele dosages) for each individual at each SNP. Element (i, j) is the MAP genotype for individual i at SNP j.

`maxpostprob` A matrix of numerics containing the variational maximum posterior probabilities for each individual at each SNP. The (i, j)th element is the variational posterior probability that individual i is genotyped correctly at SNP j.

`postprob` A three-way array of numerics. Element (i, j, k) is the variational posterior probability that individual i has genotype k-1 at SNP j.

`seq` A vector of numerics. Element j is the estimated sequencing error rate for SNP j.

`bias` A vector of numerics. Element j is the estimated allelic bias for SNP j.

`od` A vector of numerics. Element j is the estimated overdispersion parameter for SNP j.

`allele_freq` A vector of numerics. Element j is the estimated allele-frequency for SNP j.

`inbreeding` A vector of numerics. Element i is the estimated inbreeding coefficient for individual i.

`cor_mat` A symmetric matrix of numerics. Element (i, j) is the estimated _latent_ correlation between individual i and individual j.

`postmean` A matrix of numerics. Element (i, j) is the variational posterior mean for individual i at SNP j.

`postvar` A matrix of numerics. Element (i, j) is the variational posterior variance for individual i at SNP j.

`input$refmat` A matrix of numerics. The inputted `refmat`.

`input$sizemat` A matrix of numerics. The inputted `sizemat`.

`input$ploidy` The inputted `ploidy`.

`obj` The maximized variational objective.

Author(s)

David Gerard

References

- David J Balding and Richard A Nichols. Significant genetic correlations among caucasians at forensic DNA loci. Heredity, 78(6):583–589, 1997. doi: 10.1038/sj.hdy.6881750.
- Paul D Blischak, Laura S Kubatko, and Andrea D Wolfe. SNP genotyping and parameter estimation in polyploids using low-coverage sequencing data. Bioinformatics, page btx587, 2017. doi: 10.1093/bioinformatics/btx587.
- David Gerard, Luis Felipe Ventorim Ferrao, and Matthew Stephens. Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids with Messy Sequencing Data. 2018.

Examples

```
data(uitdewilligen)
mout <- mupdog(refmat = uitdewilligen$refmat,
               sizemat = uitdewilligen$sizemat,
               ploidy = uitdewilligen$ploidy,
               verbose = FALSE,
                   control = list(obj_tol = 10^-4))

## Summaries of output
plot(mout, 4)
hist(mout$bias)
hist(mout$seq)
hist(mout$od)
hist(mout$inbreeding)
hist(mout$allele_freq)

## mupdog can correctly estimate ploidy to be 4
mout2 <- mupdog(refmat = uitdewilligen$refmat,
                sizemat = uitdewilligen$sizemat,
                ploidy = 2,
                verbose = FALSE,
                control = list(obj_tol = 10^-4))

mout6 <- mupdog(refmat = uitdewilligen$refmat,
                sizemat = uitdewilligen$sizemat,
                ploidy = 6,
                verbose = FALSE,
                control = list(obj_tol = 10^-4))

mout8 <- mupdog(refmat = uitdewilligen$refmat,
                sizemat = uitdewilligen$sizemat,
                ploidy = 8,
                verbose = FALSE,
                control = list(obj_tol = 10^-4))

y <- c(mout2$obj, mout$obj, mout6$obj, mout8$obj)
x <- seq(2, 8, by = 2)
plot(x, y, type = "l", xlab = "ploidy", ylab = "objective")
```

mupout *A mupdog fit of the* `uitdewilligen` *data.*

Description

A mupdog fit of the `uitdewilligen` data.

Usage

```
mupout
```

Format

An object of class `mupdog`.

Value

See the Format Section.

Source

The raw data that this was fit to can be found in `uitdewilligen`.

See Also

`uitdewilligen`	The raw data.
`plot.mupdog`	A method to plot a `mupdog` object.
`summary.mupdog`	Calculate some summaries of a `mupdog` object.
`mupdog`	Function used to create this `mupdog` object.

`obj_for_alpha` *Objective function when updating alpha*

Description

Objective function when updating alpha

Usage

```
obj_for_alpha(mu, sigma2, alpha, rho, log_bb_dense, ploidy)
```

Arguments

`mu`	A vector. The ith element is individual i's variational posterior mean at the SNP.
`sigma2`	A vector. The ith element is individual i's variational posterior variance at the SNP.
`alpha`	The SNP's allele frequency.
`rho`	A vector. The ith element is individuals i's inbreeding coefficient.
`log_bb_dense`	A matrix of log-densities of the beta binomial. The rows index the individuals and the columns index the allele dosage.
`ploidy`	The ploidy of the species.

Value

A double. The objective when updating `alpha` in `mupdog`.

Author(s)

David Gerard

`obj_for_eps` *Objective function for updating sequencing error rate, bias, and overdispersion parameters.*

Description

Objective function for updating sequencing error rate, bias, and overdispersion parameters.

Usage

```
obj_for_eps(parvec, refvec, sizevec, ploidy, mean_bias, var_bias, mean_seq,
  var_seq, wmat, update_bias = TRUE, update_seq = TRUE, update_od = TRUE)
```

Arguments

`parvec`	A vector of length three. The first element is the sequencing error rate, the second element is the allele bias, and the third element is the overdispersion parameter.
`refvec`	A vector. The ith element is the reference count for the ith individual in the SNP.
`sizevec`	A vector. the ith element is the size count for the ith individual in the SNP/
`ploidy`	The ploidy of the species.
`mean_bias`	The prior mean of the log-bias.
`var_bias`	The prior variance of the log-bias
`mean_seq`	The prior mean of the logit sequencing error rate.
`var_seq`	The prior variance of the logit sequencing error rate.
`wmat`	The matrix of (variational) posterior probabilities for each dosage. The rows index the individuals and the columns index the dosage levels.
`update_bias`	A logical. This is not used in `obj_for_eps`, but sets the second element to `0.0` in `grad_for_eps`.
`update_seq`	A logical. This is not used in `obj_for_eps`, but sets the first element to `0.0` in `grad_for_eps`.
`update_od`	A logical. This is not used in `obj_for_eps`, but sets the third element to `0.0` in `grad_for_eps`.

Value

A double. The objective when updating `eps` in `mupdog`.

Author(s)

David Gerard

`obj_for_mu_sigma2` *Objective function when updating mu and sigma2.*

Description

Objective function when updating mu and sigma2.

Usage

```
obj_for_mu_sigma2(mu, sigma2, phifk_mat, cor_inv, log_bb_dense)
```

Arguments

`mu`	A vector, the ith element is the variational posterior mean of individual i for the SNP.
`sigma2`	A vector, the ith element is the variational posterior variance of individual i for the SNP.
`phifk_mat`	A matrix that contains the standard normal quantile of the beta-binomial cdf at dosage k for individual i. The rows index the individuals and the columns index the dosages.
`cor_inv`	The inverse of the underlying correlation matrix.
`log_bb_dense`	A matrix of log-densities of the beta binomial. The rows index the individuals and the columns index the allele dosage. Allele dosage goes from -1 to ploidy, so there are ploidy + 2 elements.

Value

A double. The objective when updating `mu` and `sigma2` in `mupdog`.

Author(s)

David Gerard

`obj_for_mu_sigma2_wrapper` *Wrapper for `obj_for_mu_sigma2` so that I can use it in `optim`.*

Description

Wrapper for `obj_for_mu_sigma2` so that I can use it in `optim`.

Usage

```
obj_for_mu_sigma2_wrapper(muSigma2, phifk_mat, cor_inv, log_bb_dense)
```

Arguments

`muSigma2`	A vector. The first half are mu and the second half are sigma2.
`phifk_mat`	A matrix that contains the standard normal quantile of the beta-binomial cdf at dosage k for individual i. The rows index the individuals and the columns index the dosages.
`cor_inv`	The inverse of the underlying correlation matrix.
`log_bb_dense`	A matrix of log-densities of the beta binomial. The rows index the individuals and the columns index the allele dosage. Allele dosage goes from -1 to ploidy, so there are ploidy + 2 elements.

Value

A double. The objective when updating `mu` and `sigma2` in `mupdog`.

Author(s)

David Gerard

`obj_for_rho` *Objective function when updating a single inbreeding coefficient.*

Description

Objective function when updating a single inbreeding coefficient.

Usage

```
obj_for_rho(rho, mu, sigma2, alpha, log_bb_dense, ploidy)
```

Arguments

`rho`	The inbreeding coefficient for the individual.
`mu`	A vector of posterior means. The jth element is the posterior mean of SNP j for the individual.
`sigma2`	A vector of posterior variances. The jth element is the posterior variance of SNP j for the individual.
`alpha`	A vector of allele frequencies. The jth element is the allele frequency for SNP j.
`log_bb_dense`	A matrix of log posterior densities. The rows index the SNPs and the columns index the dosage.
`ploidy`	The ploidy of the species.

Value

A double. The objective when updating `rho` in `mupdog`.

Author(s)

David Gerard

`obj_for_weighted_lbb` *Objective function for updating the beta-binomial genotype distribution when* `model = "bb"` *in* `flex_update_pivec`*.*

Description

Objective function for updating the beta-binomial genotype distribution when `model = "bb"` in `flex_update_pivec`.

Usage

```
obj_for_weighted_lbb(parvec, ploidy, weight_vec)
```

Arguments

`parvec`	A vector of length 2. The first term is the current mean of the underlying beta. The second term is the current overdispersion parameter.
`ploidy`	The ploidy of the species.
`weight_vec`	A vector of length `ploidy + 1` that contains the weights for each component beta-binomial.

Value

A double. The objective when updating the beta-binomial genotype distribution in `mupdog`.

Author(s)

David Gerard

`obj_for_weighted_lnorm` *Objective funtion for updating discrete normal genotype distribution when* `model = "normal"` *in* `flex_update_pivec`*.*

Description

Objective funtion for updating discrete normal genotype distribution when `model = "normal"` in `flex_update_pivec`.

Usage

```
obj_for_weighted_lnorm(parvec, ploidy, weight_vec)
```

Arguments

`parvec`	A vector of length 2. The first term is the current mean of the underlying normal. The second term is the current standard deviation (not variance) of the normal.
`ploidy`	The ploidy of the species.
`weight_vec`	A vector of length `ploidy + 1` that contains the weights for each component beta-binomial.

Value

A double. The objective when updating the normal genotype distribution in `mupdog`.

Author(s)

David Gerard

`oracle_cor` *Calculates the correlation between the true genotype and an oracle estimator.*

Description

Calculates the correlation between the oracle MAP estimator (where we have perfect knowledge about the data generation process) and the true genotype. This is a useful approximation when you have a lot of individuals.

Usage

```
oracle_cor(n, ploidy, seq, bias, od, dist)
```

Arguments

`n`	The read-depth.
`ploidy`	The ploidy of the individual.
`seq`	The sequencing error rate.
`bias`	The allele-bias.
`od`	The overdispersion parameter.
`dist`	The distribution of the alleles.

Details

To come up with `dist`, you need some additional assumptions. For example, if the population is in Hardy-Weinberg equilibrium and the allele frequency is `alpha` then you could calculate `dist` using the R code: `dbinom(x = 0:ploidy, size = ploidy, prob = alpha)`. Alternatively, if you know the genotypes of the individual's two parents are, say, `ref_count1` and `ref_count2`, then you could use the `get_q_array` function from the updog package: `get_q_array(ploidy)[ref_count1 + 1, ref_count2 + 1, ]`.

Value

The Pearson correlation between the true genotype and the oracle estimator.

Author(s)

David Gerard

References

Gerard, David, Luis Felipe Ventorim Ferrao, Antonio Augusto Franco Garcia, and Matthew Stephens. 2018. "Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids from Messy Sequencing Data." *bioRxiv*. Cold Spring Harbor Laboratory. doi:10.1101/281550.

Examples

```
## Hardy-Weinberg population with allele-frequency of 0.75.
## Moderate bias and moderate overdispersion.
## See how correlation decreases as we
## increase the ploidy.
ploidy <- 2
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
oracle_cor(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)

ploidy <- 4
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
oracle_cor(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)

ploidy <- 6
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
oracle_cor(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)
```

`oracle_cor_from_joint` *Calculate the correlation of the oracle estimator with the true genotype from the joint distribution matrix.*

Description

Calculates the correlation between the oracle MAP estimator (where we have perfect knowledge about the data generation process) and the true genotype. This is a useful approximation when you have a lot of individuals.

Usage

```
oracle_cor_from_joint(jd)
```

Arguments

`jd`	A matrix of numerics. Element (i, j) is the probability of genotype i - 1 and estimated genotype j - 1. This is usually obtained from `oracle_joint`.

Value

The Pearson correlation between the true genotype and the oracle estimator.

Author(s)

David Gerard

See Also

`oracle_joint` for getting `jd`. `oracle_cor` for not having to first calculate `jd`.

Examples

```
## Hardy-Weinberg population with allele-frequency of 0.75.
## Moderate bias and moderate overdispersion.
ploidy <- 6
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
jd <- oracle_joint(n = 100, ploidy = ploidy, seq = 0.001,
                   bias = 0.7, od = 0.01, dist = dist)
oracle_cor_from_joint(jd = jd)

## Compare to oracle_cor
oracle_cor(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)
```

`oracle_joint` *The joint probability of the genotype and the genotype estimate of an oracle estimator.*

Description

This returns the joint distribution of the true genotypes and an oracle estimator given perfect knowledge of the data generating process. This is a useful approximation when you have a lot of individuals.

Usage

```
oracle_joint(n, ploidy, seq, bias, od, dist)
```

Arguments

`n`	The read-depth.
`ploidy`	The ploidy of the individual.
`seq`	The sequencing error rate.
`bias`	The allele-bias.
`od`	The overdispersion parameter.
`dist`	The distribution of the alleles.

Details

To come up with `dist`, you need some additional assumptions. For example, if the population is in Hardy-Weinberg equilibrium and the allele frequency is `alpha` then you could calculate `dist` using the R code: `dbinom(x = 0:ploidy, size = ploidy, prob = alpha)`. Alternatively, if you know the genotypes of the individual's two parents are, say, `ref_count1` and `ref_count2`, then you could use the `get_q_array` function from the updog package: `get_q_array(ploidy)[ref_count1 + 1, ref_count2 + 1, ]`.

See the Examples to see how to reconcile the output of `oracle_joint` with `oracle_mis` and `oracle_mis_vec`.

Value

A matrix. Element (i, j) is the joint probability of estimating the genotype to be i+1 when the true genotype is j+1. That is, the estimated genotype indexes the rows and the true genotype indexes the columns. This is when using an oracle estimator.

Author(s)

David Gerard

References

Gerard, David, Luis Felipe Ventorim Ferrao, Antonio Augusto Franco Garcia, and Matthew Stephens. 2018. Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids from Messy Sequencing Data." *bioRxiv*. Cold Spring Harbor Laboratory. doi:10.1101/281550.

See Also

`oracle_plot` For visualizing the joint distribution output from `oracle_joint`.

`oracle_mis_from_joint` For obtaining the same results as `oracle_mis` directly from the output of `oracle_joint`.

`oracle_mis_vec_from_joint` For obtaining the same results as `oracle_mis_vec` directly from the output of `oracle_joint`.

`oracle_cor_from_joint` For obtaining the same results as `oracle_cor` directly from the output of `oracle_joint`.

Examples

```
## Hardy-Weinberg population with allele-frequency of 0.75.
## Moderate bias and moderate overdispersion.
ploidy <- 4
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
jd <- oracle_joint(n = 100, ploidy = ploidy, seq = 0.001,
                   bias = 0.7, od = 0.01, dist = dist)
jd

## Get same output as oracle_mis this way:
1 - sum(diag(jd))
oracle_mis(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)

## Get same output as oracle_mis_vec this way:
1 - diag(sweep(x = jd, MARGIN = 2, STATS = colSums(jd), FUN = "/"))
oracle_mis_vec(n = 100, ploidy = ploidy, seq = 0.001,
               bias = 0.7, od = 0.01, dist = dist)
```

`oracle_mis` *Calculate oracle misclassification error rate.*

Description

Given perfect knowledge of the data generating parameters, `oracle_mis` calculates the misclassification error rate, where the error rate is taken over both the data generation process and the allele-distribution. This is an ideal level of the misclassification error rate and any real method will have a larger rate than this. This is a useful approximation when you have a lot of individuals.

Usage

```
oracle_mis(n, ploidy, seq, bias, od, dist)
```

Arguments

`n`	The read-depth.
`ploidy`	The ploidy of the individual.
`seq`	The sequencing error rate.
`bias`	The allele-bias.
`od`	The overdispersion parameter.
`dist`	The distribution of the alleles.

Details

To come up with `dist`, you need some additional assumptions. For example, if the population is in Hardy-Weinberg equilibrium and the allele frequency is `alpha` then you could calculate `dist` using the R code: `dbinom(x = 0:ploidy, size = ploidy, prob = alpha)`. Alternatively, if you know the genotypes of the individual's two parents are, say, `ref_count1` and `ref_count2`, then you could use the `get_q_array` function from the updog package: `get_q_array(ploidy)[ref_count1 + 1, ref_count2 + 1, ]`.

Value

A double. The oracle misclassification error rate.

Author(s)

David Gerard

References

Gerard, David, Luis Felipe Ventorim Ferrao, Antonio Augusto Franco Garcia, and Matthew Stephens. 2018. Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids from Messy Sequencing Data." *bioRxiv*. Cold Spring Harbor Laboratory. doi:10.1101/281550.

Examples

```
## Hardy-Weinberg population with allele-frequency of 0.75.
## Moderate bias and moderate overdispersion.
## See how oracle misclassification error rates change as we
## increase the ploidy.
ploidy <- 2
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
oracle_mis(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)

ploidy <- 4
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
oracle_mis(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)

ploidy <- 6
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
oracle_mis(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)
```

`oracle_mis_from_joint` *Get the oracle misclassification error rate directly from the joint distribution of the genotype and the oracle estimator.*

Description

Get the oracle misclassification error rate directly from the joint distribution of the genotype and the oracle estimator.

Usage

```
oracle_mis_from_joint(jd)
```

Arguments

`jd`	A matrix of numerics. Element (i, j) is the probability of genotype i - 1 and estimated genotype j - 1. This is usually obtained from `oracle_joint`.

Value

A double. The oracle misclassification error rate.

Author(s)

David Gerard

See Also

`oracle_joint` for getting `jd`. `oracle_mis` for not having to first calculate `jd`.

Examples

```
## Hardy-Weinberg population with allele-frequency of 0.75.
## Moderate bias and moderate overdispersion.
ploidy <- 6
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
jd <- oracle_joint(n = 100, ploidy = ploidy, seq = 0.001,
                   bias = 0.7, od = 0.01, dist = dist)
oracle_mis_from_joint(jd = jd)

## Compare to oracle_cor
oracle_mis(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)
```

`oracle_mis_vec` *Returns the oracle misclassification rates for each genotype.*

Description

Given perfect knowledge of the data generating parameters, `oracle_mis_vec` calculates the misclassification error rate at each genotype. This differs from `oracle_mis` in that this will *not* average over the genotype distribution to get an overall misclassification error rate. That is, `oracle_mis_vec` returns a vector of misclassification error rates *conditional* on each genotype.

Usage

```
oracle_mis_vec(n, ploidy, seq, bias, od, dist)
```

Arguments

n	The read-depth.
ploidy	The ploidy of the individual.
seq	The sequencing error rate.
bias	The allele-bias.
od	The overdispersion parameter.
dist	The distribution of the alleles.

Details

This is an ideal level of the misclassification error rate and any real method will have a larger rate than this. This is a useful approximation when you have a lot of individuals.

To come up with `dist`, you need some additional assumptions. For example, if the population is in Hardy-Weinberg equilibrium and the allele frequency is `alpha` then you could calculate `dist` using the R code: `dbinom(x = 0:ploidy, size = ploidy, prob = alpha)`. Alternatively, if you know the genotypes of the individual's two parents are, say, `ref_count1` and `ref_count2`, then you could use the `get_q_array` function from the updog package: `get_q_array(ploidy)[ref_count1 + 1, ref_count2 + 1, ]`.

Value

A vector of numerics. Element i is the oracle misclassification error rate when genotyping an individual with actual genotype i + 1.

Author(s)

David Gerard

References

Gerard, David, Luis Felipe Ventorim Ferrao, Antonio Augusto Franco Garcia, and Matthew Stephens. 2018. Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids from Messy Sequencing Data." *bioRxiv*. Cold Spring Harbor Laboratory. doi:10.1101/281550.

Examples

```
## Hardy-Weinberg population with allele-frequency of 0.75.
## Moderate bias and moderate overdispersion.
ploidy <- 4
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
om <- oracle_mis_vec(n = 100, ploidy = ploidy, seq = 0.001,
                     bias = 0.7, od = 0.01, dist = dist)
om

## Get same output as oracle_mis this way:
```

```
sum(dist * om)
oracle_mis(n = 100, ploidy = ploidy, seq = 0.001,
           bias = 0.7, od = 0.01, dist = dist)
```

oracle_mis_vec_from_joint

Get the oracle misclassification error rates (conditional on true genotype) directly from the joint distribution of the genotype and the oracle estimator.

Description

Get the oracle misclassification error rates (conditional on true genotype) directly from the joint distribution of the genotype and the oracle estimator.

Usage

```
oracle_mis_vec_from_joint(jd)
```

Arguments

`jd`	A matrix of numerics. Element (i, j) is the probability of genotype i - 1 and estimated genotype j - 1. This is usually obtained from `oracle_joint`.

Value

A vector of numerics. Element i is the oracle misclassification error rate when genotyping an individual with actual genotype i + 1.

Author(s)

David Gerard

See Also

`oracle_joint` for getting `jd`. `oracle_mis_vec` for not having to first calculate `jd`.

Examples

```
## Hardy-Weinberg population with allele-frequency of 0.75.
## Moderate bias and moderate overdispersion.
ploidy <- 6
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
jd <- oracle_joint(n = 100, ploidy = ploidy, seq = 0.001,
                   bias = 0.7, od = 0.01, dist = dist)
oracle_mis_vec_from_joint(jd = jd)

## Compare to oracle_cor
```

```
oracle_mis_vec(n = 100, ploidy = ploidy, seq = 0.001,
               bias = 0.7, od = 0.01, dist = dist)
```

oracle_plot *Construct an oracle plot from the output of* oracle_joint.

Description

After obtaining the joint distribution of the true genotype with the estimated genotype from the oracle estimator using oracle_joint, you can use oracle_plot to visualize this joint distribution.

Usage

```
oracle_plot(jd)
```

Arguments

jd	A matrix containing the joint distribution of the true genotype and the oracle estimator. Usually, this is obtained by a call from oracle_joint.

Value

A ggplot object containing the oracle plot. The x-axis indexes the possible values of the estimated genotype. The y-axis indexes the possible values of the true genotype. The number in cell (i, j) is the probability that an individual will have true genotype i but is estimated to have genotype j. This is when using an oracle estimator. The cells are also color-coded by the size of the probability in each cell. At the top are listed the oracle misclassification error rate and the correlation of the true genotype with the estimated genotype. Both of these quantities may be derived from the joint distribution.

Author(s)

David Gerard

See Also

oracle_joint for obtaining jd.

Examples

```
ploidy <- 6
dist <- stats::dbinom(0:ploidy, ploidy, 0.75)
jd <- oracle_joint(n = 100, ploidy = ploidy, seq = 0.001,
                   bias = 0.7, od = 0.01, dist = dist)
pl <- oracle_plot(jd = jd)
print(pl)
```

`pbetabinom_double` *The distribution function of the betabinomial. This is generally only adviseable if q is relatively small.*

Description

The distribution function of the betabinomial. This is generally only adviseable if q is relatively small.

Usage

```
pbetabinom_double(q, size, mu, rho, log_p)
```

Arguments

`q`	A quantile.
`size`	The total number of draws.
`mu`	The mean of the beta.
`rho`	The overdispersion parameter of the beta.
`log_p`	A logical. Should return the log-probability `TRUE` or not `FALSE`?

Value

The tail-probability of the beta-binomial.

Author(s)

David Gerard

`pen_bias` *Penalty on bias parameter.*

Description

Penalty on bias parameter.

Usage

```
pen_bias(h, mu_h, sigma2_h)
```

Arguments

`h`	The current value of bias parameter. Must be greater than 0. A value of 1 means no bias.
`mu_h`	The prior mean of the log-bias parameter.
`sigma2_h`	The prior variance (not standard deviation) of the log-bias parameter. Set to to `Inf` to return `0`.

Value

A double. The default penalty on the allelic bias parameter.

Author(s)

David Gerard

`pen_seq_error` *Penalty on sequencing error rate.*

Description

Penalty on sequencing error rate.

Usage

```
pen_seq_error(eps, mu_eps, sigma2_eps)
```

Arguments

`eps`	The current value of sequencing error rate. Must be between 0 and 1.
`mu_eps`	The prior mean of the logit sequencing error rate.
`sigma2_eps`	The prior variance (not standard deviation) of the logit sequencing error rate. Set this to `Inf` to return `0`.

Value

A double. The default penalty on the sequencing error rate.

Author(s)

David Gerard

`pivec_from_segmats` *Function to get the segregation probabilities from the distributions of each component and the weights of each component.*

Description

Function to get the segregation probabilities from the distributions of each component and the weights of each component.

Usage

```
pivec_from_segmats(p1segmat, p2segmat, p1weight, p2weight)
```

Arguments

`p1segmat`	The matrix of segregation probabilities for each component for parent 1. The rows index the components and the columns index the number of alleles to segregate.
`p2segmat`	The matrix of segregation probabilities for each component for parent 2. The rows index the components and the columns index the number of alleles to segregate.
`p1weight`	A vector of weights for each component (row) of `p1segmat`.
`p2weight`	A vector of weights for each component (row) of `p2segmat`.

Value

A vector. The ith element is the probability of segregating i+1 total A alleles.

Author(s)

David Gerard

See Also

This is mostly used in `update_pp_f1`.

plot.flexdog *Draw a genotype plot from the output of* `flexdog`*.*

Description

Draw a genotype plot from the output of `flexdog`.

Usage

```
## S3 method for class 'flexdog'
plot(x, use_colorblind = TRUE, ...)
```

Arguments

`x`	A `flexdog` object.
`use_colorblind`	Should we use a colorblind-safe palette (`TRUE`) or not (`FALSE`)? `TRUE` is only allowed if the ploidy is less than or equal to 6.
`...`	Not used.

Details

On a genotype plot, the x-axis contains the counts of the non-reference allele and the y-axis contains the counts of the reference allele. The dashed lines are the expected counts (both reference and alternative) given the sequencing error rate and the allele-bias. The plots are color-coded by the maximum-a-posterior genotypes. Transparency is proportional to the maximum posterior probability for an individual's genotype. Thus, we are less certain of the genotype of more transparent individuals.

Value

A `ggplot` object for the genotype plot.

Author(s)

David Gerard

See Also

`plot_geno` The underlying plotting function.

`flexdog` Creates a `flexdog` object.

`plot.mupdog` *Draw a genotype plot from the output of* `mupdog`.

Description

A wrapper for `plot_geno`. This will create a genotype plot for a single SNP.

Usage

```
## S3 method for class 'mupdog'
plot(x, index, use_colorblind = TRUE, ...)
```

Arguments

`x`	A mupdog object.
`index`	The column number of the gene to plot.
`use_colorblind`	Should we use a colorblind-safe palette (`TRUE`) or not (`FALSE`)? `TRUE` is only allowed if the ploidy is less than or equal to 6.
`...`	Not used.

Details

On a genotype plot, the x-axis contains the counts of the non-reference allele and the y-axis contains the counts of the reference allele. The dashed lines are the expected counts (both reference and alternative) given the sequencing error rate and the allele-bias. The plots are color-coded by the maximum-a-posterior genotypes. Transparency is proportional to the maximum posterior probability for an individual's genotype. Thus, we are less certain of the genotype of more transparent individuals.

Value

A `ggplot` object for the genotype plot.

Author(s)

David Gerard

See Also

`plot_geno` The underlying plotting function.

`mupdog` Creates a `mupdog` object.

Examples

```
data("mupout")
plot(mupout, 4)
```

plot_geno *Make a genotype plot.*

Description

The x-axis is the counts of the non-reference allele, and the y-axis is the counts of the reference allele. Transparency is controlled by the `maxpostprob` vector.

Usage

```
plot_geno(refvec, sizevec, ploidy, p1ref = NULL, p1size = NULL,
  p2ref = NULL, p2size = NULL, geno = NULL, seq = 0, bias = 1,
  maxpostprob = NULL, p1geno = NULL, p2geno = NULL,
  use_colorblind = TRUE)
```

Arguments

`refvec`	A vector of non-negative integers. The number of reference reads observed in the individuals
`sizevec`	A vector of positive integers. The total number of reads in the individuals.
`ploidy`	A non-negative integer. The ploidy of the species.
`p1ref`	A vector of non-negative integers. The number of reference reads observed in parent 1 (if the individuals are all siblings).
`p1size`	A vector of positive integers. The total number of reads in parent 1 (if the individuals are all siblings).
`p2ref`	A vector of non-negative integers. The number of reference reads observed in parent 2 (if the individuals are all siblings).
`p2size`	A vector of positive integers. The total number of reads in parent 2 (if the individuals are all siblings).
`geno`	The individual genotypes.
`seq`	The sequencing error rate.
`bias`	The bias parameter.
`maxpostprob`	A vector of the posterior probabilities of being at the modal genotype.
`p1geno`	Parent 1's genotype.
`p2geno`	Parent 2's genotype.
`use_colorblind`	A logical. Should we use a colorblind safe palette (`TRUE`), or not (`FALSE`)? Only allowed if `ploidy <= 6`.

Details

If parental genotypes are provided (`p1geno` and `p2geno`) then they will be colored the same as the offspring. Since they are often hard to see, a small black dot will also indicate their position.

Value

A ggplot object for the genotype plot.

Author(s)

David Gerard

Examples

```
data("snpdat")
refvec  <- snpdat$counts[snpdat$snp == "SNP1"]
sizevec <- snpdat$size[snpdat$snp == "SNP1"]
ploidy  <- 6
plot_geno(refvec = refvec, sizevec = sizevec, ploidy = ploidy)
```

---

post_prob — *Variational posterior probability of having* dosage *A alleles when the ploidy is* ploidy*, the allele frequency is* alpha*, the individual-specific overdispersion parameter is* rho*, the variational mean is* mu*, and the variational variance is* sigma2.

Description

Variational posterior probability of having dosage A alleles when the ploidy is ploidy, the allele frequency is alpha, the individual-specific overdispersion parameter is rho, the variational mean is mu, and the variational variance is sigma2.

Usage

```
post_prob(dosage, ploidy, mu, sigma2, alpha, rho)
```

Arguments

dosage	The number of A alleles.
ploidy	The ploidy of the individual.
mu	The variational mean.
sigma2	The variational variance (not standard devation).
alpha	The allele frequency.
rho	The individual's overdispersion parameter.

Value

The posterior probability of having dosage A alleles.

Author(s)

David

pp_brent_obj *Objective function when doing Brent's method in update_pp_f1 when one parent only has two mixing components.*

Description

Objective function when doing Brent's method in update_pp_f1 when one parent only has two mixing components.

Usage

```
pp_brent_obj(firstmixweight, probmat, pvec, weight_vec, alpha)
```

Arguments

firstmixweight	The mixing weight of the first component.
probmat	The rows index the components and the columns index the segregation amount. Should only have two rows.
pvec	The distribution of the other parent.
weight_vec	The weights for each element.
alpha	The mixing weight on the uniform component.

Value

The objective value, as calculated by taking a convolution using convolve of the mixing distribution and pvec, then putting that probability distribution through f1_obj.

Author(s)

David Gerard

qbetabinom_double *The quantile function of the beta-binomial distribution parameterized by mean and overdispersion parameter.*

Description

The quantile function of the beta-binomial distribution parameterized by mean and overdispersion parameter.

Usage

```
qbetabinom_double(p, size, mu, rho)
```

Arguments

`p`	The lower tail probability.
`size`	The total number of draws.
`mu`	The mean of the beta.
`rho`	The overdispersion parameter of the beta.

Value

The quantile of the beta-binomial.

Author(s)

David Gerard

`rbetabinom_int` *One draw from the beta-binomial distribution parameterized by mean and overdispersion parameter.*

Description

One draw from the beta-binomial distribution parameterized by mean and overdispersion parameter.

Usage

```
rbetabinom_int(size, mu, rho)
```

Arguments

`size`	The total number of draws.
`mu`	The mean of the beta.
`rho`	The overdispersion parameter of the beta.

Value

A random sample from the beta-binomial.

Author(s)

David Gerard

rflexdog — *Simulate GBS data from the flexdog likelihood.*

Description

This will take a vector of genotypes and a vector of total read-counts, then generate a vector of reference counts. To get the genotypes, you could use rgeno.

Usage

```
rflexdog(sizevec, geno, ploidy, seq = 0.005, bias = 1, od = 0.001)
```

Arguments

sizevec	A vector of total read-counts for the individuals.
geno	A vector of genotypes for the individuals. I.e. the number of reference alleles each individual has.
ploidy	The ploidy of the species.
seq	The sequencing error rate.
bias	The bias parameter. Pr(a read after selected) / Pr(A read after selected).
od	The overdispersion parameter. See the Details of the rho variable in betabinom.

Value

A vector the same length as sizevec. The ith element is the number of reference counts for individual i.

Author(s)

David Gerard

References

Gerard, David, Luis Felipe Ventorim Ferrao, Antonio Augusto Franco Garcia, and Matthew Stephens. 2018. "Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids from Messy Sequencing Data." *bioRxiv*. Cold Spring Harbor Laboratory. doi:10.1101/281550.

See Also

rgeno for a way to generate genotypes of individuals. rbetabinom for how we generate the read-counts.

Examples

```
set.seed(1)
n       <- 100
ploidy  <- 6

## Generate the genotypes of individuals from an F1 population,
## where the first parent has 1 copy of the reference allele
## and the second parent has two copies of the reference
## allele.
genovec <- rgeno(n = n, ploidy = ploidy, model = "f1",
                 p1geno = 1, p2geno = 2)

## Get the total number of read-counts for each individual.
## Ideally, you would take this from real data as the total
## read-counts are definitely not Poisson.
sizevec <- stats::rpois(n = n, lambda = 200)

## Generate the counts of reads with the reference allele
## when there is a strong bias for the reference allele
## and there is no overdispersion.
refvec  <- rflexdog(sizevec = sizevec, geno = genovec,
                    ploidy = ploidy, seq = 0.001,
                    bias = 0.5, od = 0)

## Plot the simulated data using plot_geno.
plot_geno(refvec = refvec, sizevec = sizevec,
          ploidy = ploidy, seq = 0.001, bias = 0.5)
```

`rgeno` *Simulate individual genotypes from one of the supported* `flexdog` *models.*

Description

This will simulate genotypes of a sample of individuals drawn from one of the populations supported by `flexdog`. See the details of `flexdog` for the models allowed.

Usage

```
rgeno(n, ploidy, model = c("hw", "bb", "norm", "ash", "f1", "s1", "f1pp",
  "s1pp", "flex", "uniform"), allele_freq = NULL, od = NULL,
  p1geno = NULL, p2geno = NULL, mode = NULL, pivec = NULL, mu = NULL,
  sigma = NULL, p1_pair_weights = NULL, p2_pair_weights = NULL)
```

Arguments

`n`	The number of observations.
`ploidy`	The ploidy of the species.

`model`	What form should the prior take? See Details in `flexdog`.
`allele_freq`	If `model = "hw"`, then this is the allele frequency of the population. For any other model, this should be `NULL`.
`od`	If `model = "bb"`, then this is the overdispersion parameter of the beta-binomial distribution. See `betabinom` for details. For any other model, this should be `NULL`.
`p1geno`	Either the first parent's genotype if `model = "f1"` (or `model = "f1pp"`), or the only parent's genotype if `model = "s1"` (or `model = "s1pp"`). For any other model, this should be `NULL`.
`p2geno`	The second parent's genotype if `model = "f1"` (or `model = "f1pp"`). For any other model, this should be `NULL`.
`mode`	If `model = "ash"`, this is the center of the distribution. This should be a non-integer value (so the mode is either the floor or the ceiling of `mode`). For any other model, this should be `NULL`.
`pivec`	A vector of probabilities. If `model = "ash"`, then this represents the mixing proportions of the discrete uniforms. If `model = "flex"`, then element `i` is the probability of genotype `i - 1`. For any other model, this should be `NULL`.
`mu`	If `model = "norm"`, this is the mean of the normal. For any other model, this should be `NULL`.
`sigma`	If `model = "norm"`, this is the standard deviation of the normal. For any other model, this should be `NULL`.
`p1_pair_weights`	The mixing weights for the bivalent pairs output in `get_bivalent_probs` at the `lvec` level of `p1geno`. If `model = "f1pp"` then this is for the first parent. If `model = "s1pp"`, then this is for the only parent. This should be `NULL` for all other values of `model`.
`p2_pair_weights`	If `model = "s1pp"`, these are the mixing weights for the bivalent pairs output in `get_bivalent_probs` at the `lvec` level of `p2geno` for the second parent. This should be `NULL` for all other values of `model`.

Details

The allowable variable values of `allele_freq`, `od`, `p1geno`, `p2geno`, `pivec`, and `mode` varies based on the value of `model`. If `model = "ash"`, then only `mode` and `pivec` can be non-`NULL`. If `model = "flex"` then only `pivec` can be non-`NULL`. If `model = "hw"`, then only `allele_freq` can be non-`NULL`. If `model = "f1"` then only `p1geno` and `p2geno` can be non-`NULL`. If `model = "s1"` then only `p1geno` can be non-`NULL`. If `model = "uniform"`, then none of the above variables can be non-`NULL`. If `model = "bb"`, then only `allele_freq`, and `od` can be non-`NULL`. If `model == "norm"`, then only `mu` and `sigma` can be non-`NULL`.

Value

A vector of length `n` with the genotypes of the sampled individuals.

Author(s)

David Gerard

References

Gerard, David, Luis Felipe Ventorim Ferrao, Antonio Augusto Franco Garcia, and Matthew Stephens. 2018. "Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids from Messy Sequencing Data." *bioRxiv*. Cold Spring Harbor Laboratory. doi:10.1101/281550.

Examples

```
## F1 Population where parent 1 has 1 copy of the referenc allele
## and parent 2 has 4 copies of the reference allele.
ploidy <- 6
rgeno(n = 10, ploidy = ploidy, model = "f1", p1geno = 1, p2geno = 4)

## A population in Hardy-Weinberge equilibrium with an
## allele frequency of 0.75
rgeno(n = 10, ploidy = ploidy, model = "hw", allele_freq = 0.75)
```

`snpdat` *GBS data from Shirasawa et al (2017)*

Description

Contains counts of reference alleles and total read counts from the GBS data of Shirasawa et al (2017) for the three SNP's used as examples in Gerard, Ferrao, and Stephens (2017).

Usage

```
snpdat
```

Format

A `tibble` with 419 rows and 4 columns:

id The identification label of the individuals.

snp The SNP label.

counts The number of read-counts that support the reference allele.

size The total number of read-counts at a given SNP.

Value

A `tibble`. See the Format Section.

Source

`http://sweetpotato-garden.kazusa.or.jp/`

References

Shirasawa, Kenta, Masaru Tanaka, Yasuhiro Takahata, Daifu Ma, Qinghe Cao, Qingchang Liu, Hong Zhai et al. "A high-density SNP genetic map consisting of a complete set of homologous groups in autohexaploid sweetpotato (Ipomoea batatas)." Scientific Reports 7 (2017). DOI: 10.1038/srep44207

Gerard, David, Luis Felipe Ventorim Ferrão, and Matthew Stephens. 2017. "Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids with Messy Sequencing Data." Overleaf Preprint.

---

summary.mupdog *Provides some summaries from the output of* mupdog.

Description

Returns mean-dosage for each individual at each SNP and the SNP-specific distribution of MAP dosages. The mean-dosage in particular might be of interest for downstream analyses.

Usage

```
## S3 method for class 'mupdog'
summary(object, ...)
```

Arguments

object	A mupdog object.
...	Not used.

Value

A list with the following elements:

- `freq` A matrix with the frequency of the dosages (rows) for each SNP (columns).
- `mean_dosage` The posterior mean dosage of an individual (rows) for each SNP (columns).

Author(s)

David Gerard

Examples

```
data(mupout)
msum <- summary(mupout)
msum$freq[, 1:5]
boxplot(msum$mean_dosage ~ mupout$map_dosage,
        xlab = "MAP Dosage", ylab = "Mean Dosage")
```

`uitdewilligen` *Subset of individuals and SNPs from Uitdewilligen et al (2013).*

Description

A list containing a matrix of reference counts, a matrix of total counts, and the ploidy level (4) of the species. This is a subset of the data from Uitdewilligen et al (2013).

Usage

```
uitdewilligen
```

Format

A list containing three objects. Two matrices and a numeric scalar:

refmat A matrix of read counts containing the reference allele. The rows index the individuals and the columns index the SNPs.

sizemat A matrix of the total number of read counts. The rows index the individuals and the columns index the SNPs.

ploidy The ploidy level of the species (just 4).

Value

A list. See the Format Section.

Source

`https://doi.org/10.1371/journal.pone.0062355`

References

Uitdewilligen, J. G., Wolters, A. M. A., Bjorn, B., Borm, T. J., Visser, R. G., & van Eck, H. J. (2013). A next-generation sequencing method for genotyping-by-sequencing of highly heterozygous autotetraploid potato. PLoS One, 8(5), e62355.

See Also

`mupout`: a mupdog fit of these data.

`uni_em` *EM algorithm to fit weighted ash objective.*

Description

Solves the following optimization problem

$$\max_{\pi} \sum_k w_k \log(\sum_j \pi_j \ell_j k).$$

It does this using a weighted EM algorithm.

Usage

```
uni_em(weight_vec, lmat, pi_init, lambda, itermax, obj_tol)
```

Arguments

`weight_vec`	A vector of weights. Each element of `weight_vec` corresponds to a column of `lmat`.
`lmat`	A matrix of inner weights. The columns are the "individuals" and the rows are the "classes."
`pi_init`	The initial values of `pivec`. Each element of `pi_init` corresponds to a row of `lmat`.
`lambda`	The penalty on the pi's. Should be greater than 0 and really really small.
`itermax`	The maximum number of EM iterations to take.
`obj_tol`	The objective stopping criterion.

Value

A vector of numerics. The update of `pivec` in `flexdog_full`.

Author(s)

David Gerard

`uni_em_const` *EM algorithm to fit weighted ash objective with a uniform mixing component.*

Description

Solves the following optimization problem

$$\max_{\pi} \sum_k w_k \log(\alpha/(K+1) + (1-\alpha)\sum_j \pi_j \ell_j k).$$

It does this using a weighted EM algorithm.

Usage

```
uni_em_const(weight_vec, lmat, pi_init, alpha, lambda, itermax, obj_tol)
```

Arguments

`weight_vec`	A vector of weights. Each element of `weight_vec` corresponds to a column of `lmat`.
`lmat`	A matrix of inner weights. The columns are the "individuals" and the rows are the "classes."
`pi_init`	The initial values of `pivec`. Each element of `pi_init` corresponds to a row of `lmat`.
`alpha`	The mixing weight for the uniform component. This should be small (say, less tahn 10^-3).
`lambda`	A vector of penalties on the pi's (corresponding to the rows of `lmat`). This can either be of length 1, in which case the same penalty is applied to each of the pi's. Or it can be the same length of `pivec`, in which case a different penalty is applied to each of the pi's. Larger penalties generally increase the value of the pi's, not shrink them.
`itermax`	The maximum number of EM iterations to take.
`obj_tol`	The objective stopping criterion.

Value

A vector of numerics. The update of `pivec` in `flexdog_full`.

Author(s)

David Gerard

`uni_obj` *Objective function optimized by* `uni_em`*.*

Description

Objective function optimized by `uni_em`.

Usage

```
uni_obj(pivec, weight_vec, lmat, lambda)
```

Arguments

`pivec`	The current parameters.
`weight_vec`	A vector of weights. Each element of `weight_vec` corresponds to a column of `lmat`.
`lmat`	A matrix of inner weights. The columns are the "individuals" and the rows are the "classes."
`lambda`	The penalty on the pi's. Should be greater than 0 and really really small.

Value

The objective optimized by `uni_em` during that separate unimodal EM algorithm.

Author(s)

David Gerard

`uni_obj_const` *Objective function optimized by* `uni_em_const`*.*

Description

Objective function optimized by `uni_em_const`.

Usage

```
uni_obj_const(pivec, alpha, weight_vec, lmat, lambda)
```

Arguments

`pivec`	The current parameters.
`alpha`	The mixing weight for the uniform component. This should be small (say, less tahn 10^-3).
`weight_vec`	A vector of weights. Each element of `weight_vec` corresponds to a column of `lmat`.
`lmat`	A matrix of inner weights. The columns are the "individuals" and the rows are the "classes."
`lambda`	A vector of penalties on the pi's (corresponding to the rows of `lmat`). This can either be of length 1, in which case the same penalty is applied to each of the pi's. Or it can be the same length of `pivec`, in which case a different penalty is applied to each of the pi's. Larger penalties generally increase the value of the pi's, not shrink them.

Value

The objective optimized by `uni_em_const` during that separate unimodal EM algorithm.

Author(s)

David Gerard

`update_dr` *Same as `update_pp_f1` but I exclusively use the EM (instead of also Brent's method), and I allow for priors on the mixing proportions.*

Description

Same as `update_pp_f1` but I exclusively use the EM (instead of also Brent's method), and I allow for priors on the mixing proportions.

Usage

```
update_dr(weight_vec, model = c("f1pp", "f1ppdr"), control)
```

Arguments

`weight_vec`	`colSums(wik_mat)` from `flexdog`. This is the sum of current posterior probabilities of each individual having genotype k.
`model`	The model to assume.
`control`	A list of anything else needed to be passed. E.g. if `model = "ash"`, then `inner_weights` needs to be passed through `control` (see `get_inner_weights` for how to get this matrix).

Value

A list with the following elements:

`p1_pair_weights` A list with the mixing weights for the bivalent components of parent 1.

`p2_pair_weights` A list with the mixing weights for the bivalent components of parent 2.

`obj` The maximized objective.

`p1geno` The estimated genotype for parent 1.

`p2geno` The estimated genotype for parent 2.

`pivec` The estimated genotype distribution for the offspring.

See Also

`update_pp_f1`

`update_pp_f1` *Function to update the parameters in the preferential pairing F1 model.*

Description

Function to update the parameters in the preferential pairing F1 model.

Usage

```
update_pp_f1(weight_vec, control)
```

Arguments

`weight_vec`	`colSums(wik_mat)` from `flexdog`. This is the sum of current posterior probabilities of each individual having genotype k.
`control`	A list of anything else needed to be passed. E.g. if `model = "ash"`, then `inner_weights` needs to be passed through `control` (see `get_inner_weights` for how to get this matrix).

Value

A list with the following elements:

`p1_pair_weights` A list with the mixing weights for the bivalent components of parent 1.

`p2_pair_weights` A list with the mixing weights for the bivalent components of parent 2.

`obj` The maximized objective.

`p1geno` The estimated genotype for parent 1.

`p2geno` The estimated genotype for parent 2.

`pivec` The estimated genotype distribution for the offspring.

Author(s)

David Gerard

See Also

update_dr

---

update_pp_s1 — *Same as update_pp_f1 but only allow s1.*

Description

Same as update_pp_f1 but only allow s1.

Usage

```
update_pp_s1(weight_vec, control)
```

Arguments

weight_vec	colSums(wik_mat) from flexdog. This is the sum of current posterior probabilities of each individual having genotype k.
control	A list of anything else needed to be passed. E.g. if model = "ash", then inner_weights needs to be passed through control (see get_inner_weights for how to get this matrix).

Value

A list with the following elements:

p1_pair_weights A list with the mixing weights for the bivalent components of parent 1.

p2_pair_weights A list with the mixing weights for the bivalent components of parent 2.

obj The maximized objective.

p1geno The estimated genotype for parent 1.

p2geno The estimated genotype for parent 2.

pivec The estimated genotype distribution for the offspring.

See Also

update_pp_f1

`update_R` *Update the underlying correlation matrix.*

Description

Update the underlying correlation matrix.

Usage

```
update_R(postmean, postvar)
```

Arguments

`postmean`	The matrix of posterior means. The rows index the individuals and the columns index the SNPs.
`postvar`	The matrix of posterior variances. The rows index the individuals and the columns index the SNPs.

Value

A symmetric matrix of numerics. The update of the underlying correlation matrix.

Author(s)

David Gerard

`updog` `updog` *Flexible Genotyping for Polyploids*

Description

Implements empirical Bayes approaches to genotype polyploids from next generation sequencing data while accounting for allelic bias, overdispersion, and sequencing error. The main function is `flexdog`, which allows the specification of many different genotype distributions. An experimental function that takes into account varying levels of relatedness is implemented in `mupdog`. Also provided are functions to simulate genotypes (`rgeno`) and read-counts (`rflexdog`), as well as functions to calculate oracle genotyping error rates (`oracle_mis`) and correlation with the true genotypes (`oracle_cor`). These latter two functions are useful for read depth calculations. Run `browseVignettes(package = "updog")` in R for example usage.

Details

The package is named updog for "Using Parental Data for Offspring Genotyping" because we originally developed the method for full-sib populations, but it works now for more general populations.

Our best competitor is probably the `fitPoly` package, which you can check out at `https://cran.r-project.org/package=fitPoly`. Though, we think that updog returns better calibrated measures of uncertainty when you have next-generation sequencing data.

If you find a bug or want an enhancement, please submit an issue at `http://github.com/dcgerard/updog/issues`.

updog **Functions**

- `flexdog` The main function that fits an empirical Bayes approach to genotype polyploids from next generation sequencing data.
- `mupdog` An experimental approach to genotype autopolyploids that accounts for varying levels of relatedness between the individuals in the sample.
- `rgeno` simulate the genotypes of a sample from one of the models allowed in `flexdog`.
- `rflexdog` Simulate read-counts from the `flexdog` model.
- `plot.flexdog` Plotting the output of `flexdog`.
- `plot.mupdog` Plotting the output of `mupdog`.
- `oracle_joint` The joint distribution of the true genotype and an oracle estimator.
- `oracle_plot` Visualize the output of `oracle_joint`.
- `oracle_mis` The oracle misclassification error rate (Bayes rate).
- `oracle_cor` Correlation between the true genotype and the oracle estimated genotype.

updog **Datasets**

- `snpdat` A small example dataset for using `flexdog`.
- `uitdewilligen` A small example dataset for using `mupdog`.
- `mupout` The output from fitting `mupdog` to `uitdewilligen`.

Author(s)

David Gerard

References

Gerard, David, Luis Felipe Ventorim Ferrao, Antonio Augusto Franco Garcia, and Matthew Stephens. 2018. "Harnessing Empirical Bayes and Mendelian Segregation for Genotyping Autopolyploids from Messy Sequencing Data." *bioRxiv*. Cold Spring Harbor Laboratory. doi:10.1101/281550.

wem *Generalized version of uni_em.*

Description

Generalized version of uni_em.

Usage

```
wem(weight_vec, lmat, pi_init, lambda, itermax, obj_tol)
```

Arguments

weight_vec	A vector of weights. Each element of weight_vec corresponds to a column of lmat.
lmat	A matrix of inner weights. The columns are the "individuals" and the rows are the "classes."
pi_init	The initial values of pivec. Each element of pi_init corresponds to a row of lmat.
lambda	The penalty on the pi's. Should be greater than 0 and really really small.
itermax	The maximum number of EM iterations to take.
obj_tol	The objective stopping criterion.

Value

A vector of numerics. The update of pivec in flexdog_full.

Author(s)

David Gerard

See Also

uni_em for a description of the optimization problem.

Examples

```
set.seed(2)
n <- 3
p <- 5
lmat <- matrix(stats::runif(n * p), nrow = n)
weight_vec <- seq_len(p)
pi_init <- stats::runif(n)
pi_init <- pi_init / sum(pi_init)
wem(weight_vec = weight_vec,
    lmat       = lmat,
    pi_init    = pi_init,
    lambda     = 0,
    itermax    = 100,
    obj_tol    = 10^-6)
```

`xi_double` *Adjusts allele dosage* `p` *by the sequencing error rate* `eps` *and the allele bias* `h`.

Description

Adjusts allele dosage `p` by the sequencing error rate `eps` and the allele bias `h`.

Usage

```
xi_double(p, eps, h)
```

Arguments

`p`	The allele dosage.
`eps`	The sequencing error rate.
`h`	The allele bias.

Value

The probability of a reference read adjusted by both the allele bias and the sequencing error rate.

Author(s)

David Gerard

`xi_fun` *Adjusts allele dosage* `p` *by the sequencing error rate* `eps` *and the allele bias* `h`.

Description

Adjusts allele dosage `p` by the sequencing error rate `eps` and the allele bias `h`.

Usage

```
xi_fun(p, eps, h)
```

Arguments

`p`	A vector of allele dosages.
`eps`	The sequencing error rate. Must either be of length 1 or the same length as p.
`h`	The allele bias. Must either be of length 1 or the same length as p.

Value

A vector of prababilities of the reference read adjusted by both the sequencing error rate and the allele bias.

Author(s)

David Gerard

Index