Package ‘segtest’

June 30, 2025

Title Tests for Segregation Distortion in Polyploids

Version 2.0.0

Description Provides tests for segregation distortion in F1 polyploid populations under different assumptions of meiosis. These tests can account for double reduction, partial preferential pairing, and genotype uncertainty through the use of genotype likelihoods. Parallelization support is provided. Details of these methods are described in Gerard et al. (2025a) <doi:10.1007/s00122-025-04816-z> and Gerard et al. (2025b) <doi:10.1101/2025.06.23.661114>. Part of this material is based upon work supported by the National Science Foundation under Grant No. 2132247. The opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views of the National Science Foundation.

License GPL (>= 3)

BugReports `https://github.com/dcgerard/segtest/issues`

URL `https://dcgerard.github.io/segtest/`

Encoding UTF-8

RoxygenNote 7.3.2

Biarch true

Depends R (>= 3.5)

Imports doFuture, doRNG, foreach, future, iterators, minqa, nloptr, Rcpp, updog

Suggests knitr, polymapR, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

LinkingTo Rcpp, RcppArmadillo

LazyData true

NeedsCompilation yes

Author David Gerard [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9450-5023>),
Mira Thakkar [aut],
Guilherme da Silva Pereira [ctb] (ORCID:
<https://orcid.org/0000-0002-7106-8630>),
NSF DBI 2132247 [fnd]
(https://www.nsf.gov/awardsearch/showAward?AWD_ID=2132247)

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

Repository CRAN

Date/Publication 2025-06-30 18:40:02 UTC

Contents

`beta_bounds` *Bounds on the distortion at simplex loci caused by double reduction.*

Description

The frequency of (nullplex, simplex, duplex) gametes is (.5 + beta, .5 - 2 * beta, beta). This function returns the upper bound on beta under two models.

Usage

```
beta_bounds(ploidy, model = c("ces", "prcs"))
```

Arguments

`ploidy`	The ploidy
`model`	Either complete equational segregation (`"ces"`) (Mather, 1935) or pure random chromatid segregation `"prcs"`) (Haldane, 1930). See also Huang et al. (2019).

Details

Returns the upper bound on the probability of a gamete with a genotype of 2 when the parent has a genotype of 1. This is based on two models. The upper bound from complete equational separation is higher than the upper bound from the pure random chromatid segregation. See Huang et al (2019) for a description of these models.

Value

The upper bound on beta.

Author(s)

David Gerard

References

- Huang, K., Wang, T., Dunn, D. W., Zhang, P., Cao, X., Liu, R., & Li, B. (2019). Genotypic frequencies at equilibrium for polysomic inheritance under double-reduction. *G3: Genes, Genomes, Genetics*, 9(5), 1693-1706. doi:10.1534/g3.119.400132

Examples

```
beta_bounds(4)
beta_bounds(6)
beta_bounds(8)
beta_bounds(10)
```

---

chisq_g *Chi Square test when genotypes are known*

Description

This chi-squared test is run under the assumption of no double reduction and no preferential pairing.

Usage

```
chisq_g(x, g1, g2)

chisq_g4(x, g1, g2)
```

Arguments

x	Vector of observed genotype counts
g1	Parent 1's genotype
g2	Parent 2's genotype

Value

A list containing the chi-squared statistic, degrees of freedom, and p-value.

Functions

- `chisq_g4()`: Alias for chisq_g, for backwards compatibility.

Author(s)

Mira Thakkar and David Gerard

Examples

```
x <- c(1, 2, 4, 3, 0)
g1 <- 2
g2 <- 2
chisq_g(x, g1, g2)

x <- c(10, 25, 10, 0, 0)
g1 <- 1
g2 <- 1
chisq_g(x, g1, g2)
```

`chisq_gl` *Chi-Sq for GL*

Description

Calculates the MLE genotype and runs a chi-squared test assuming no double reduction and no preferential pairing.

Usage

```
chisq_gl(gl, g1, g2)

chisq_gl4(gl, g1, g2)
```

Arguments

`gl`	A matrix of offspring genotype log-likelihoods. The rows index the individuals and the columns index the possible genotypes. So `gl[i, k]` is the offspring genotype log-likelihood for individual i and genotype k-1.
`g1`	The first parent's genotype.
`g2`	The second parent's genotype.

Value

A list containing the chi-squared statistic, degrees of freedom, and p-value.

Functions

- `chisq_gl4()`: Alias for chisq_gl, for backwards compatibility.

Author(s)

Mira Thakkar and David Gerard

Examples

```
## null sim
set.seed(1)
g1 <- 2
g2 <- 2
gl <- simf1gl(n = 25, g1 = g1, g2 = g2, alpha = 0, xi2 = 1/3)
chisq_gl(gl = gl, g1 = g1, g2 = g2)
```

drbounds *Upper bounds on double reduction rates.*

Description

Provides the upper bounds on the double reduction rates based on the formulas in Huang et al. (2019). There are two upper bounds provided. The upper bound from complete equational separation is higher than the upper bound from the pure random chromatid segregation.

Usage

```
drbounds(ploidy, model = c("ces", "prcs"))
```

Arguments

`ploidy`	The ploidy
`model`	Either complete equational segregation (`"ces"`) (Mather, 1935) or pure random chromatid segregation `"prcs"`) (Haldane, 1930). See also Huang et al. (2019).

Value

A vector of length `floor(ploidy / 4)` containing the upper bounds on the rates of double reduction. The `i`the element is the upper bound on the probability that there are `i` pairs of identical by double reduction alleles in a gamete.

Author(s)

David Gerard

References

- Haldane, J. B. S. (1930). Theoretical genetics of autopolyploids. *Journal of genetics*, 22, 359-372. doi:10.1007/BF02984197
- Huang, K., Wang, T., Dunn, D. W., Zhang, P., Cao, X., Liu, R., & Li, B. (2019). Genotypic frequencies at equilibrium for polysomic inheritance under double-reduction. *G3: Genes, Genomes, Genetics*, 9(5), 1693-1706. doi:10.1534/g3.119.400132
- Mather, K. (1935). Reductional and equational separation of the chromosomes in bivalents and multivalents. *Journal of genetics*, 30, 53-78. doi:10.1007/BF02982205

Examples

```
drbounds(4)
drbounds(6)
drbounds(8)
drbounds(10)
```

`em_li` *EM algorithm from Li (2011)*

Description

EM algorithm to estimate prior genotype probabilities from genotype likelihoods.

Usage

```
em_li(B, itermax = 100L, eps = 1e-05)
```

Arguments

`B`	Matrix of genotype log-likelihoods. The rows index the individuals and the columns index the genotypes.
`itermax`	The maximum number of iterations.
`eps`	The stopping criteria.

Value

A vector of log prior probabilities for each genotype.

Author(s)

David Gerard

References

- Li, H. (2011). A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, 27(21), 2987-2993. doi:10.1093/bioinformatics/btr509

Examples

```
# Simulate some data
set.seed(1)
gl <- simgl(nvec = c(3, 2, 4, 1, 2))
# Run em
lprob <- em_li(B = gl)
# Exponentiate to get probabilities
prob <- exp(c(lprob))
prob
```

gamfreq *Gamete frequencies under a generalized model*

Description

Returns the gamete frequencies for autopolyploids, allopolyploids, and segmental allopolyploids, accounting for the effects of double reduction and partial preferential pairing.

Usage

```
gamfreq(
  g,
  ploidy,
  gamma = NULL,
  alpha = NULL,
  beta = NULL,
  type = c("mix", "polysomic"),
  add_dr = TRUE
)
```

Arguments

`g`	Parent genotype.
`ploidy`	Parent ploidy. Should be even, and between 2 and 20 (inclusive). Let me know if you need the ploidy to be higher. I can update the package really easily.
`gamma`	The mixture proportions for the pairing configurations. The proportions are in the same order the configurations in `seg`. See Gerard et al (2018) for details on pairing configurations.
`alpha`	The double reduction rate(s) (if using). Defaults to 0's.
`beta`	The double reduction adjustment for simplex markers if `type = "mix"` and `add_dr = TRUE`. Assumed to be 0 by default.
`type`	Either `"mix"`, meaning a mixture model of pairing configurations, or `"polysomic"` for polysomic inheritance.
`add_dr`	A logical. If `type = "polysomic"`, then the double double reduction rate (`"alpha"`) will be used no matter the value of `add_dr`, so set `alpha = 0` if you don't want it. But if `type = "mix"` then we will incorporate double reduction only in simplex markers (where it matters the most, and where preferential pairing does not operate).

Value

The vector of gamete frequencies. Element `i` is the probability a gamete has genotype `i - 1`.

Models

If type = "polysomic", then the gamete frequencies correspond to those of Huang et al (2019). Those formulas are for general multiallelic loci, so see also Appendix G of Gerard (2022) for special case of biallelic loci. The relevant parameter is alpha, a vector of length floor(ploidy / 4), where alpha[[i]] is the probability that there are i pairs of double reduced alleles in a gamete. The theoretical upper bound on alpha is given in drbounds().

If type = "mix" and add_dr = FALSE, then the gamete frequencies correspond to the pairing configuration model of Gerard et al (2018). This model states that the gamete frequencies are a convex combination of the disomic inheritance frequencies. The weights of this convex combination are provided in the gamma parameter. The total number of disomic segregation patterns is given by n_pp_mix(). The order of these segregation patterns used is the order in seg.

The model for type = "mix" and add_dr = TRUE is the same as for type = "mix" and add_dr = FALSE *except* at parental simplex loci. At such loci, there are no effects of preferential pairing, and so the option add_dr = TRUE allows for the effects of double reduction at simplex loci. The relevant parameter here is beta. The first three gamete frequencies at simplex loci are c(0.5 + beta, 0.5 - 2 * beta, beta), and the rest are 0. The upper bound on beta for two different models are given by beta_bounds().

Author(s)

David Gerard

References

- Gerard, D. (2023). Double reduction estimation and equilibrium tests in natural autopolyploid populations. *Biometrics*, 79(3), 2143-2156. doi:10.1111/biom.13722
- Gerard, D., Ferrão, L. F. V., Garcia, A. A. F., & Stephens, M. (2018). Genotyping polyploids from messy sequencing data. *Genetics*, 210(3), 789-807. doi:10.1534/genetics.118.301468
- Huang, K., Wang, T., Dunn, D. W., Zhang, P., Cao, X., Liu, R., & Li, B. (2019). Genotypic frequencies at equilibrium for polysomic inheritance under double-reduction. *G3: Genes, Genomes, Genetics*, 9(5), 1693-1706. doi:10.1534/g3.119.400132

Examples

```
## Various duplex models
gamfreq(g = 2, ploidy = 4, gamma = c(0, 1), type = "mix")
gamfreq(g = 2, ploidy = 4, gamma = c(1, 0), type = "mix")
gamfreq(g = 2, ploidy = 4, gamma = c(0.5, 0.5), type = "mix")
gamfreq(g = 2, ploidy = 4, alpha = 0, type = "polysomic")
gamfreq(g = 2, ploidy = 4, alpha = 1/6, type = "polysomic")

## Various simplex models
gamfreq(g = 1, ploidy = 4, beta = 1/24, gamma = 1, type = "mix", add_dr = TRUE)
gamfreq(g = 1, ploidy = 4, alpha = 1/6, type = "polysomic")
gamfreq(g = 1, ploidy = 4, gamma = 1, type = "mix", add_dr = FALSE)
gamfreq(g = 1, ploidy = 4, alpha = 0, type = "polysomic")
```

gcount_to_gvec *Converts genotype counts to genotype vectors.*

Description

Converts genotype counts to genotype vectors.

Usage

```
gcount_to_gvec(gcount)
```

Arguments

`gcount`	The vector of genotype counts.

Value

A vector of length `sum(gcount)`, containing `gcount[1]` copies of `0`, `gcount[2]` copies of `1`, `gcount[3]` copies of `2`, etc.

Author(s)

David Gerard

See Also

`gvec_to_gcount()`

Examples

```
gcount <- c(1, 2, 3, 0, 5)
gcount_to_gvec(gcount = gcount)
```

gf_freq *Genotype frequencies of an F1 population under a generalized model.*

Description

Genotype frequencies of an F1 population under a generalized model.

Usage

```
gf_freq(
  p1_g,
  p1_ploidy,
  p1_gamma = NULL,
  p1_alpha = NULL,
  p1_beta = NULL,
  p1_type = c("mix", "polysomic"),
  p1_add_dr = TRUE,
  p2_g,
  p2_ploidy,
  p2_gamma = NULL,
  p2_alpha = NULL,
  p2_beta = NULL,
  p2_type = c("mix", "polysomic"),
  p2_add_dr = TRUE,
  pi = 0,
  nudge = sqrt(.Machine$double.eps)
)
```

Arguments

`p1_g`, `p1_ploidy`, `p1_gamma`, `p1_alpha`, `p1_beta`, `p1_type`, `p1_add_dr`	The first parent's version of the parameters in `gamfreq()`.
`p2_g`, `p2_ploidy`, `p2_gamma`, `p2_alpha`, `p2_beta`, `p2_type`, `p2_add_dr`	The second parent's version of the parameters in `gamfreq()`.
`pi`	The proportion of outliers.
`nudge`	Zeros go to nudge

Value

A vector of genotype frequencies. Element `i` is the probability and offspring has genotype `i - 1`.

Author(s)

David Gerard

See Also

`gamfreq()`.

Examples

```
q <- gf_freq(
  p1_g = 2,
  p1_ploidy = 4,
  p1_gamma = c(0.1, 0.9),
  p1_type = "mix",
  p2_g = 2,
  p2_ploidy = 6,
  p2_alpha = 0.1,
  p2_type = "polysomic",
  pi = 0.05)
```

gvec_to_gcount *Inverse function of* gcount_to_gvec().

Description

Inverse function of gcount_to_gvec().

Usage

```
gvec_to_gcount(gvec, ploidy = 4)
```

Arguments

gvec	The vector of genotypes. gvec[i] is the genotype for individual i.
ploidy	The ploidy of the species.

Value

A vector of counts. Element k is the number of individuals with genotype k-1.

Author(s)

David Gerard

See Also

gcount_to_gvec()

Examples

```
gvec <- c(1, 2, 3, 2, 3, 1, 4, 0, 1, 0, 0, 1, 0, 0)
gvec_to_gcount(gvec = gvec)
```

is_valid_2 *Tests if the two parameter model is valid*

Description

There is a dependence on the bounds of two-parameter model. This function returns TRUE if those bounds are satisfied and FALSE otherwise.

Usage

```
is_valid_2(dr, pp, drbound = 1/6)
```

Arguments

`dr`	The double reduction rate.
`pp`	The preferential pairing parameter.
`drbound`	The maximum double reduction rate possible.

Value

`TRUE` if the model is valid, `FALSE` otherwise.

Author(s)

David Gerard

Examples

```
TOL <- 1e-6
is_valid_2(dr = 1/6, pp = 1/3, drbound = 1/6) # Valid
is_valid_2(dr = 1/6, pp = 1/3 - TOL, drbound = 1/6) # Not valid
is_valid_2(dr = 1/6, pp = 1/3 + TOL, drbound = 1/6) # Not valid
```

iter.array *Iterator over array*

Description

Iterator over array

Usage

```
## S3 method for class 'array'
iter(obj, by = 1, recycle = FALSE, ...)
```

Arguments

`obj`	An array.
`by`	The dimension to iterate over.
`recycle`	Should the iterator reset?
`...`	not used

Value

An iterator. This is an S3 arrayiter object, used in conjunction with nextElem to iterate over one index of an array.

Author(s)

David Gerard

See Also

`nextElem.arrayiter()`

Examples

```
glist <- multidog_to_g(
  mout = ufit,
  ploidy = 4,
  type = "all_gl",
  p1 = "indigocrisp",
  p2 = "sweetcrisp")
g <- iterators::iter(glist$g, by = 3)
head(iterators::nextElem(g))
head(iterators::nextElem(g))
head(iterators::nextElem(g))
```

`like_gknown_2` *Likelihood under three parameter model when genotypes are known*

Description

This is under the two parameter model.

Usage

```
like_gknown_2(x, alpha, xi1, xi2, g1, g2, log_p = TRUE, pen = 0)
```

Arguments

`x`	A vector of length 5. `x[i]` is the count of individuals with genotype `i-1`.
`alpha`	The double reduction rate.
`xi1`	The preferential pairing parameter of parent 1.
`xi2`	The preferential pairing parameter of parent 2.
`g1`	Parent 1's genotype.
`g2`	Parent 2's genotype.
`log_p`	A logical. Should we return the log likelihood or not?
`pen`	A tiny penalty to help with numerical stability

Value

The (log) likelihood.

Author(s)

David Gerard

Examples

```
x <- c(1, 4, 5, 3, 1)
alpha <- 0.01
xi1 <- 0.5
xi2 <- 0.3
g1 <- 1
g2 <- 2
like_gknown_2(
  x = x,
  alpha = alpha,
  xi1 = xi1,
  xi2 = xi2,
  g1 = g1,
  g2 = g2)
```

`like_gknown_3` *Likelihood under three parameter model when genotypes are known*

Description

This is under the three parameter model.

Usage

```
like_gknown_3(x, tau, beta, gamma1, gamma2, g1, g2, log_p = TRUE, pen = 0)
```

Arguments

`x`	A vector of length 5. `x[i]` is the count of individuals with genotype `i-1`.
`tau`	The probability of quadrivalent formation.
`beta`	The probability of double reduction given quadrivalent formation.
`gamma1`	The probability of AA_aa pairing for parent 1.
`gamma2`	The probability of AA_aa pairing for parent 2.
`g1`	Parent 1's genotype.
`g2`	Parent 2's genotype.
`log_p`	A logical. Should we return the log likelihood or not?
`pen`	A tiny penalty to help with numerical stability

Value

The (log) likelihood.

Author(s)

David Gerard

Examples

```
x <- c(1, 4, 5, 3, 1)
tau <- 0.5
beta <- 0.1
gamma1 <- 0.5
gamma2 <- 0.3
g1 <- 1
g2 <- 2
like_gknown_3(
  x = x,
  tau = tau,
  beta = beta,
  gamma1 = gamma1,
  gamma2 = gamma2,
  g1 = g1,
  g2 = g2)
```

`like_glpknown_2` *Likelihood under three parameter model when using offspring genotypes likelihoods but parent genotypes are known.*

Description

This is under the two parameter model.

Usage

```
like_glpknown_2(gl, alpha, xi1, xi2, g1, g2, log_p = TRUE)
```

Arguments

`gl`	The matrix of genotype likelihoods of the offspring. Rows index The individuals, columns index the genotypes.
`alpha`	The double reduction rate.
`xi1`	The preferential pairing parameter of parent 1.
`xi2`	The preferential pairing parameter of parent 2.
`g1`	Parent 1's genotype.
`g2`	Parent 2's genotype.
`log_p`	A logical. Should we return the log likelihood or not?

Value

The (log) likelihood of the two parameter model when using genotype likelihoods.

Author(s)

David Gerard

Examples

```
g1 <- 1
g2 <- 0
gl <- simf1gl(
  n = 25,
  g1 = g1,
  g2 = g2,
  rd = 10,
  alpha = 0,
  xi1 = 1/3,
  xi2 = 1/3)
like_glpknown_2(
  gl = gl,
  alpha = 0.01,
  xi1 = 0.5,
  xi2 = 0.3,
  g1 = g1,
  g2 = g2,
  log_p = TRUE)
```

`like_glpknown_3` *Likelihood under three parameter model when using offspring genotypes likelihoods but parent genotypes are known.*

Description

This is under the three parameter model.

Usage

```
like_glpknown_3(gl, tau, beta, gamma1, gamma2, g1, g2, log_p = TRUE)
```

Arguments

`gl`	The matrix of genotype likelihoods of the offspring. Rows index The individuals, columns index the genotypes.
`tau`	The probability of quadrivalent formation.
`beta`	The probability of double reduction given quadrivalent formation.
`gamma1`	The probability of AA_aa pairing for parent 1.
`gamma2`	The probability of AA_aa pairing for parent 2.
`g1`	Parent 1's genotype.
`g2`	Parent 2's genotype.
`log_p`	A logical. Should we return the log likelihood or not?

Value

The (log) likelihood of the three parameter model when using genotype likelihoods.

Author(s)

David Gerard

Examples

```
g1 <- 1
g2 <- 0
gl <- simf1gl(
  n = 25,
  g1 = g1,
  g2 = g2,
  rd = 10,
  alpha = 0,
  xi1 = 1/3,
  xi2 = 1/3)
like_glpknown_3(
  gl = gl,
  tau = 1/2,
  beta = 1/12,
  gamma1 = 1/3,
  gamma2 = 1/3,
  g1 = g1,
  g2 = g2,
  log_p = TRUE)
```

llike_li *Objective function for* em_li()

Description

Objective function for em_li()

Usage

```
llike_li(B, lpivec)
```

Arguments

B	The log-likelihood matrix. Rows are individuals columns are genotypes.
lpivec	The log prior vector.

Value

The log-likelihood of a vector of genotype frequencies when using genotype likelihoods. This is from Li (2011).

Author(s)

David Gerard

References

- Li, H. (2011). A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, 27(21), 2987-2993. doi:10.1093/bioinformatics/btr509

Examples

```
# Simulate some data
set.seed(1)
gl <- simgl(nvec = c(3, 2, 4, 1, 2))
# Log-likelihood at given log-priors
prob <- c(0.1, 0.2, 0.4, 0.2, 0.1)
lprob <- log(prob)
llike_li(B = gl, lpivec = lprob)
```

`lrt_men_g4` *Likelihood ratio test for segregation distortion with known genotypes*

Description

This will run a likelihood ratio test using the genotypes of an F1 population of tetraploids for the null of Mendelian segregation (accounting for double reduction and preferential pairing) against the alternative of segregation distortion. This is when the genotypes are assumed known.

Usage

```
lrt_men_g4(
  x,
  g1,
  g2,
  drbound = 1/6,
  pp = TRUE,
  dr = TRUE,
  alpha = 0,
  xi1 = 1/3,
  xi2 = 1/3
)
```

Arguments

`x`	A vector of genotype counts. `x[i]` is the number of offspring with genotype `i-1`.
`g1`	The genotype of parent 1.
`g2`	The genotype of parent 2.
`drbound`	The maximum rate of double reduction. A default of 1/6 is provided, which is the rate under the complete equational segregation model of meiosis.
`pp`	A logical. Should we account for preferential pairing (`TRUE`) or not (`FALSE`)?
`dr`	A logical. Should we account for double reduction (`TRUE`) or not (`FALSE`)?
`alpha`	If `dr = FALSE`, this is the known rate of double reduction.
`xi1`	If `pp = FALSE`, this is the known preferential pairing parameter of parent 1.
`xi2`	If `pp = FALSE`, this is the known preferential pairing parameter of parent 2.

Value

A list with the following elements

`statistic` The log-likelihood ratio test statistic.

`df` The degrees of freedom.

`p_value` The p-value.

`alpha` The estimated double reduction rate.

`xi1` The estimated preferential pairing parameter of parent 1.

`xi2` The estimated preferential pairing parameter of parent 2.

Impossible genotypes

Some offspring genotype combinations are impossible given the parental genotypes. If these impossible genotypes combinations show up, we return a p-value of 0, a log-likelihood ratio statistic of Infinity, and missing values for all other return items. The impossible genotypes are:

`g1 = 0 && g2 = 0` Only offspring genotypes of 0 are possible.

`g1 = 4 && g2 = 4` Only offspring genotypes of 4 are possible.

`g1 = 0 && g2 = 4 || g1 == 4 && g2 == 0` Only offspring genotypes of 2 are possible.

`g1 = 0 && g2 %in% c(1, 2, 3) || g1 = %in% c(1, 2, 3) && g2 == 0` Only offspring genotypes of 0, 1, and 2 are possible.

`g1 = 4 && g2 %in% c(1, 2, 3) || g1 = %in% c(1, 2, 3) && g2 == 4` Only offspring genotypes of 2, 3, and 4 are possible.

Unidentified parameters

When `g1 = 2` or `g2 = 2` (or both), the model is not identified and those estimates (`alpha`, `xi1`, and `xi2`) are meaningless. Do NOT interpret them.

The estimate of `alpha` (double reduction rate) IS identified as long as at least one parent is simplex, and no parent is duplex. However, the estimates of the double reduction rate have extremely high variance.

Author(s)

David Gerard

Examples

```
set.seed(100)
gf <- offspring_gf_2(alpha = 1/12, xi1 = 0.2, xi2 = 0.6, p1 = 1, p2 = 0)
x <- offspring_geno(gf = gf, n = 100)
lrt_men_g4(x = x, g1 = 1, g2 = 0)
```

`lrt_men_gl4` *Likelihood ratio test using genotype likelihoods.*

Description

This will run a likelihood ratio test using the genotypes of an F1 population of tetraploids for the null of Mendelian segregation (accounting for double reduction and preferential pairing) against the alternative of segregation distortion. This is when genotype uncertainty is accounted for through genotype likelihoods.

Usage

```
lrt_men_gl4(
  gl,
  g1 = NULL,
  g2 = NULL,
  drbound = 1/6,
  pp = TRUE,
  dr = TRUE,
  alpha = 0,
  xi1 = 1/3,
  xi2 = 1/3
)
```

Arguments

`gl`	The genotype log-likelihoods. The rows index the individuals and the columns index the genotypes.
`g1`	Either parent 1's genotype, or parent 1's genotype log-likelihoods.
`g2`	Either parent 2's genotype, or parent 2's genotype log-likelihoods.
`drbound`	The upper bound on the double reduction rate.
`pp`	Is (partial) preferential pairing possible (`TRUE`) or not (`FALSE`)?
`dr`	Is double reduction possible (`TRUE`) or not (`FALSE`)?
`alpha`	If `dr = FALSE`, this is the known rate of double reduction.
`xi1`	If `pp = FALSE`, this is the known preferential pairing parameter of parent 1.
`xi2`	If `pp = FALSE`, this is the known preferential pairing parameter of parent 2.

Value

A list with the following elements

`statistic` The log-likelihood ratio test statistic.

`df` The degrees of freedom.

`p_value` The p-value.

`alpha` The estimated double reduction rate.

`xi1` The estimated preferential pairing parameter of parent 1.

`xi2` The estimated preferential pairing parameter of parent 2.

Unidentified parameters

When `g1 = 2` or `g2 = 2` (or both), the model is not identified and those estimates (`alpha`, `xi1`, and `xi2`) are meaningless. Do NOT interpret them.

The estimate of `alpha` (double reduction rate) IS identified as long as at least one parent is simplex, and no parent is duplex. However, the estimates of the double reduction rate have extremely high variance.

Author(s)

David Gerard

Examples

```
## null simulation
set.seed(1)
g1 <- 2
g2 <- 2
gl <- simf1gl(n = 25, g1 = g1, g2 = g2, alpha = 1/12, xi2 = 1/2)

## LRT when parent genotypes are known.
lrt_men_gl4(gl = gl, g1 = g1, g2 = g2)

## LRT when parent genotypes are not known
lrt_men_gl4(gl = gl)

## Alternative simulation
gl <- simgl(nvec = rep(5, 5))
lrt_men_gl4(gl = gl, g1 = g1, g2 = g2)
```

`multidog_to_g` *Converts multidog output to a format usable for seg_multi() and multi_lrt()*

Description

Converts multidog output to a format usable for seg_multi() and multi_lrt()

Usage

```
multidog_to_g(
  mout,
  ploidy,
  type = c("off_gl", "all_gl", "off_g", "all_g"),
  p1 = NULL,
  p2 = NULL
)
```

Arguments

`mout`	The output of `multidog()`.
`ploidy`	The ploidy.
`type`	`"off_gl"` Genotype likelihoods of offspring but not parents. This is the typical choice if you used the "f1", "f1pp", "s1", or "s1pp" options when genotyping.

"all_gl" Genotype likelihoods of offspring and parents. This is only done if you did *not* use the "f1", "f1pp", "s1", or "s1pp" options when genotyping. If this is the case, then you need to specify which individuals are the parents.

"off_g" Genotypes, assuming that they are known. You used the "f1", "f1pp", "s1", or "s1pp" option when genotyping.

"all_g" Genotypes, assuming that they are known. You did *not* use the "f1", "f1pp", "s1", or "s1pp" option when genotyping. If this is the case, then you need to specify which individuals are the parents.

p1	The first (or only) parent name if using type = "all_gl" or type = "all_g".
p2	The second parent name if using type = "all_gl" or type = "all_g". Omit if you used the "s1" or "s1pp" models when genotyping.

Value

A list with the following elements

g Either a matrix of counts, where the columns index the genotype and the rows index the loci (type = "all_g" or type = "off_g"). Or an array of genotype (natural) log-likelihoods where the rows index the loci, the columns index the individuals, and the slices index the genotypes (type = "all_gl" or type = "off_gl").

p1 Either a vector of known parental genotypes (type = "off_gl", type = "all_g" or type = "off_g"). Or a matrix of genotype (natural) log-likelihoods where the rows index the loci and the columns index the genotypes (type = "all_gl").

p2 Either a vector of known parental genotypes (type = "off_gl", type = "all_g" or type = "off_g"). Or a matrix of genotype (natural) log-likelihoods where the rows index the loci and the columns index the genotypes (type = "all_gl"). This will be NULL if you (i) used "s1" or "s1pp" models in updog and used either type = "off_g" or type = "off_gl" or (ii) used type = "all_g" or type = "all_gl" and only specified p1 but not p2.

Author(s)

David Gerard

Examples

```
multidog_to_g(
  mout = ufit,
  ploidy = 4,
  type = "all_g",
  p1 = "indigocrisp",
  p2 = "sweetcrisp")
multidog_to_g(
  mout = ufit,
  ploidy = 4,
  type = "all_gl",
  p1 = "indigocrisp",
  p2 = "sweetcrisp")
multidog_to_g(mout = ufit2, ploidy = 4, type = "off_g")
multidog_to_g(mout = ufit2, ploidy = 4, type = "off_gl")
multidog_to_g(mout = ufit3, ploidy = 4, type = "off_g")
multidog_to_g(mout = ufit3, ploidy = 4, type = "off_gl")
```

---

multi_lrt — *Parallelized likelihood ratio test for segregation distortion.*

Description

Uses the `future` package to implement parallelization support for the likelihood ratio tests for segregation distortion. This function only works for tetraploids, and cannot account for outliers. For higher ploidies and more functionality, see `seg_multi()`.

Usage

```
multi_lrt(
  g,
  p1,
  p2,
  drbound = 1/6,
  pp = TRUE,
  dr = TRUE,
  alpha = 0,
  xi1 = 1/3,
  xi2 = 1/3,
  nullprop = FALSE
)
```

Arguments

`g`	One of two inputs • A matrix of genotype counts. The rows index the loci and the columns index the genotypes. • An array of genotype log-likelihoods. The rows index the loci, the columns index the individuals, and the slices index the genotypes. Log-likelihoods are base e (natural log).
`p1`	One of three inputs • A vector of parent 1's genotypes. • A matrix of parent 1's genotype log-likelihoods. The rows index the loci and the columns index the genotypes. Logs are in base e (natural log). • `NULL` (only supported when using genotype likelihoods for the offspring)
`p2`	One of three inputs • A vector of parent 1's genotypes. • A matrix of parent 1's genotype log-likelihoods. The rows index the loci and the columns index the genotypes. Logs are in base e (natural log).

- `NULL` (only supported when using genotype likelihoods for the offspring)

`drbound`	The upper bound on the double reduction rate.
`pp`	Is (partial) preferential pairing possible (`TRUE`) or not (`FALSE`)?
`dr`	Is double reduction possible (`TRUE`) or not (`FALSE`)?
`alpha`	If `dr = FALSE`, this is the known rate of double reduction.
`xi1`	If `pp = FALSE`, this is the known preferential pairing parameter of parent 1.
`xi2`	If `pp = FALSE`, this is the known preferential pairing parameter of parent 2.
`nullprop`	Should we return the null proportions (`TRUE`) or not (`FALSE`)?

Value

A data frame with the following elements:

`statistic` The likelihood ratio test statistic

`p_value` The p-value of the likelihood ratio test.

`df` The degrees of freedom of the test.

`alpha` The MLE of the double reduction rate. Do not use for real work.

`xi1` The MLE of the first parent's partial preferential pairing parameter. Do not use for real work.

`xi2` The MLE of the second parent's partial preferential pairing parameter. Do not use for real work.

`p1` (Estimate of) the first parent's genotype.

`p2` (Estimate of) the second parent's genotype.

`snp` The name of the SNP.

Parallel Computation

The `multi_lrt()` function supports parallel computing. It does so through the future package.

You first specify the evaluation plan with `plan()` from the `future` package. On a local machine, this is typically just `future::plan(future::multisession, workers = nc)` where `nc` is the number of workers you want. You can find the maximum number of possible workers with `availableCores()`. You then run `multi_lrt()`, then shut down the workers with `future::plan(future::sequential)`.

Author(s)

David Gerard

See Also

- `lrt_men_g4()` Single locus LRT for segregation distortion when genotypes are known.
- `lrt_men_gl4()` Single locus LRT for segregation distortion when using genotype likelihoods.

Examples

```
## Assuming genotypes are known (typically a bad idea)
glist <- multidog_to_g(
  mout = ufit,
  ploidy = 4,
  type = "all_g",
  p1 = "indigocrisp",
  p2 = "sweetcrisp")
p1_1 <- glist$p1
p2_1 <- glist$p2
g_1 <- glist$g
multi_lrt(g = g_1, p1 = p1_1, p2 = p2_1)

## Using genotype likelihoods (typically a good idea)
glist <- multidog_to_g(
  mout = ufit,
  ploidy = 4,
  type = "all_gl",
  p1 = "indigocrisp",
  p2 = "sweetcrisp")
p1_2 <- glist$p1
p2_2 <- glist$p2
g_2 <- glist$g
multi_lrt(g = g_2, p1 = p1_2, p2 = p2_2)

## Offspring genotype likelihoods and parent genotypes known
multi_lrt(g = g_2, p1 = p1_1, p2 = p2_1)

## Offspring genotype likelihoods and no information on parent genotypes
multi_lrt(g = g_2, p1 = NULL, p2 = NULL)

## Parallel computing is supported through the future package
# future::plan(future::multisession, workers = 2)
# multi_lrt(g = g_2, p1 = p1_2, p2 = p2_2)
# future::plan(future::sequential)
```

`nextElem.arrayiter` *Next element in an array*

Description

This is applied to an `arrayiter` object to obtain the next sub-array along one of the dimensions.

Usage

```
## S3 method for class 'arrayiter'
nextElem(obj, ...)
```

Arguments

`obj`	An arrayiter object
`...`	not used

Value

The next sub-array.

Author(s)

David Gerard

See Also

`iter.array()`

Examples

```
glist <- multidog_to_g(
  mout = ufit,
  ploidy = 4,
  type = "all_gl",
  p1 = "indigocrisp",
  p2 = "sweetcrisp")
g <- iterators::iter(glist$g, by = 3)
head(iterators::nextElem(g))
head(iterators::nextElem(g))
head(iterators::nextElem(g))
```

`n_pp_mix` *Number of mixture components*

Description

The number of disomic inheritance patterns for a given ploidy and a given parental dosage. See also `seg` for the list of all possible disomic inheritance patterns for even ploidies up to 20.

Usage

```
n_pp_mix(g, ploidy)
```

Arguments

`g`	parent genotype
`ploidy`	parent ploidy

Value

The number of mixture components.

Examples

```
n_pp_mix(g = 0, ploidy = 4)
n_pp_mix(g = 1, ploidy = 4)
n_pp_mix(g = 2, ploidy = 4)
n_pp_mix(g = 3, ploidy = 4)
n_pp_mix(g = 4, ploidy = 4)

n_pp_mix(g = 0, ploidy = 6)
n_pp_mix(g = 1, ploidy = 6)
n_pp_mix(g = 2, ploidy = 6)
n_pp_mix(g = 3, ploidy = 6)
n_pp_mix(g = 4, ploidy = 6)
n_pp_mix(g = 5, ploidy = 6)
n_pp_mix(g = 6, ploidy = 6)
```

`offspring_geno` *Simulates genotypes given genotype frequencies.*

Description

Takes as input the offspring genotype frequencies and a sample size and returns simulated genotypes.

Usage

```
offspring_geno(gf, n)
```

Arguments

`gf`	Vector of offspring genotype frequencies
`n`	Sample size

Value

Simulated genotypes

Author(s)

Mira Thakkar

Examples

```
set.seed(1)
gf <- offspring_gf_2(alpha = 1/6, xi1 = 1/3, xi2 = 1/3, p1 = 2, p2 = 3)
offspring_geno(gf = gf, n = 10)
```

`offspring_gf_2` *Calculates offspring genotype frequencies under the two-parameter model.*

Description

Calculates offspring genotype frequencies under the two-parameter model.

Usage

```
offspring_gf_2(alpha, xi1, xi2 = xi1, p1, p2)
```

Arguments

alpha	The double reduction rate
xi1	The preferential pairing parameter of the first parent.
xi2	The preferential pairing parameter of the second parent.
p1	The first parent's genotype
p2	The second parent's genotype

Value

Offspring genotype frequencies

Author(s)

Mira Thakkar

Examples

```
alpha <- 1/6
xi1 <- 1/3
xi2 <- 1/3
p1 <- 2
p2 <- 3
offspring_gf_2(alpha = alpha, xi1 = xi1, xi2 = xi2, p1 = p1, p2 = p2)
```

`offspring_gf_3` *Calculates offspring genotype frequencies under the three-parameter model.*

Description

Calculates offspring genotype frequencies under the three-parameter model.

Usage

```
offspring_gf_3(tau, beta, gamma1, gamma2 = gamma1, p1, p2)
```

Arguments

`tau`	Probability of quadrivalent formation
`beta`	Probability of double reduction given quadrivalent formation
`gamma1`	Probability of AA_aa pairing in parent 1
`gamma2`	Probability of AA_aa pairing in parent 2
`p1`	The first parent's genotype
`p2`	The second parent's genotype

Value

Offspring genotype frequencies

Author(s)

David Gerard

Examples

```
offspring_gf_3(
  tau = 1/2,
  beta = 1/6,
  gamma1 = 1/3,
  gamma2 = 1/3,
  p1 = 1,
  p2 = 2)
```

otest_g — *Jointly tests for segregation distortion and number of incompatible genotypes*

Description

This is experimental. I haven't tested it out in lots of scenarios yet.

Usage

```
otest_g(
  x,
  g1,
  g2,
  pbad = 0.03,
  drbound = 1/6,
  pp = TRUE,
  dr = TRUE,
  alpha = 0,
  xi1 = 1/3,
  xi2 = 1/3
)
```

Arguments

`x`	A vector of genotype counts. `x[i]` is the number of offspring with genotype `i-1`.
`g1`	The genotype of parent 1.
`g2`	The genotype of parent 2.
`pbad`	The upper bound on the number of bad genotypes
`drbound`	The maximum rate of double reduction. A default of 1/6 is provided, which is the rate under the complete equational segregation model of meiosis.
`pp`	A logical. Should we account for preferential pairing (`TRUE`) or not (`FALSE`)?
`dr`	A logical. Should we account for double reduction (`TRUE`) or not (`FALSE`)?
`alpha`	If `dr = FALSE`, this is the known rate of double reduction.
`xi1`	If `pp = FALSE`, this is the known preferential pairing parameter of parent 1.
`xi2`	If `pp = FALSE`, this is the known preferential pairing parameter of parent 2.

Details

Here, we test if the compatible genotypes are consistent with F1 populations and separately test that the number of incompatible genotypes isn't too large (less than 3 percent by default). This is the strategy the polymapR software uses. But we use a Bonferroni correction to combine these tests (minimum of two times the p-values), while they just multiply the p-values together. So our approach accounts for double reduction and preferential pairing, while also controlling the family-wise error rate.

Value

A list with the following elements

`statistic` The log-likelihood ratio test statistic.

`df` The degrees of freedom.

`p_value` The Bonferroni corrected p-value.

`p_lrt` The p-value of the LRT.

`p_binom` The p-value of the one-sided binomial test.

`alpha` The estimated double reduction rate.

`xi1` The estimated preferential pairing parameter of parent 1.

`xi2` The estimated preferential pairing parameter of parent 2.

Author(s)

David Gerard

Examples

```
# Run a test where genotypes 0, 1, and 2 are possible
x <- c(10, 10, 4, 0, 5)
otest_g(x = x, g1 = 1, g2 = 0)

# polymapR's multiplication and the Bonferroni differ
df <- expand.grid(p1 = seq(0, 1, length.out = 20), p2 = seq(0, 1, length.out = 20))
df$polymapr <- NA
df$bonferroni <- NA
for (i in seq_len(nrow(df))) {
  df$polymapr[[i]] <- df$p1[[i]] * df$p2[[i]]
  df$bonferroni[[i]] <- 2 * min(c(df$p1[[i]], df$p2[[i]], 0.5))
}
graphics::plot(df$polymapr, df$bonferroni)
```

`polymapr_test` *Run segregation distortion tests as implemented in the polymapR package.*

Description

The polymapR package tests for segregation distortion by iterating through all possible forms of disomic or polysomic inheritance from either parent, tests for concordance of the offspring genotypes using a chi-squared test, and returns the largest p-value. It sometimes chooses a different p-value based on other heuristics. They also sometimes return NA. When `type = "segtest"`, we only look at patterns of the given parent genotypes, choosing the largest p-value. When `type = "polymapR"`, we return what they use via their heuristics.

Usage

```
polymapr_test(x, g1 = NULL, g2 = NULL, type = c("segtest", "polymapR"))
```

Arguments

x	Either a vector of genotype counts, or a matrix of genotype posteriors where the rows index the individuals and the columns index the genotypes.
g1	Parent 1's genotype.
g2	Parent 2's genotype.
type	Either my implementation which approximates that of polymapR (`"segtest"`) or the implementation through polymapR (`"polymapR"`). Note that polymapR needs to be installed for `type = "polymapR"`.

Value

A list with the following elements:

p_value The p-value of the test.

bestfit The genotype frequencies of the best fit model.

frq_invalid The frequency of invalid genotypes.

p_invalid The p-value of the invalid proportion.

Author(s)

David Gerard

See Also

`checkF1()`.

Examples

```
g1 <- 0
g2 <- 1
x <- c(4, 16, 0, 0, 0)
polymapr_test(x = x, g1 = g1, g2 = g2, type = "segtest")
polymapr_test(x = x, g1 = g1, g2 = g2, type = "polymapR")
```

po_gl *Generate genotype likelihoods from offspring genotypes.*

Description

Takes as input (i) the parent genotypes, (ii) the offspring genotype frequency, (iii) sequencing error rate, (iv) read depth, (v) bias, and (vi) overdispersion. It returns genotype likelihoods.

Usage

```
po_gl(
  genovec,
  ploidy,
  p1_geno = NULL,
  p2_geno = NULL,
  rd = 10,
  seq = 0.01,
  bias = 1,
  od = 0.01
)
```

Arguments

`genovec`	Offspring genotypes. `genovec[i]` is the dosage for individual i.
`ploidy`	Ploidy
`p1_geno`	Parent 1 genotype
`p2_geno`	Parent 2 genotype
`rd`	Read depth. Lower is more uncertain.
`seq`	Sequencing error rate. Higher means more uncertain.
`bias`	Bias. 1 means no bias.
`od`	Overdispersion. Typical value is like 0.01. Higher means more uncertain.

Value

Genotype likelihoods

Author(s)

Mira Thakkar

Examples

```
set.seed(1)
po_gl(genovec = c(1, 2, 1, 1, 3), p1_geno = 2, p2_geno = 2, ploidy = 4)
```

`pvec_tet_2` *Tetraploid gamete frequencies of gametes when one parent's genotype is known*

Description

This is under the two parameter model.

Usage

```
pvec_tet_2(alpha, xi, ell)
```

Arguments

`alpha`	The double reduction rate
`xi`	The preferential pairing parameter
`ell`	The parental genotype

Value

The gamete genotype frequencies

Author(s)

Mira Thakkar and David Gerard

Examples

```
alpha <- 1/6
xi <- 1/3
pvec_tet_2(alpha = alpha, xi = xi, ell = 0)
pvec_tet_2(alpha = alpha, xi = xi, ell = 1)
pvec_tet_2(alpha = alpha, xi = xi, ell = 2)
pvec_tet_2(alpha = alpha, xi = xi, ell = 3)
pvec_tet_2(alpha = alpha, xi = xi, ell = 4)
```

`pvec_tet_3` *Tetraploid gamete frequencies of gametes when one parent's genotype is known*

Description

This is under the three parameter model.

Usage

```
pvec_tet_3(tau, beta, gamma, ell)
```

Arguments

`tau`	Probability of quadrivalent formation
`beta`	Probability of double reduction given quadrivalent formation
`gamma`	Probability of AA/aa pairing given bivalent formation
`ell`	The parent genotype

Value

The gamete genotype frequencies

Author(s)

David Gerard

Examples

```
pvec_tet_3(tau = 0.5, beta = 0.1, gamma = 0.5, ell = 2)
```

`seg` *Disomic and polysomic segregation patterns*

Description

Gamete frequencies for all possible disomic and polysomic segregation patterns for even ploidies 2 through 20. If you need higher ploidy levels, let me know and I'll update it (it's very easy).

Usage

```
seg
```

Format

A data frame with the following columns

`ploidy` The ploidy of the parent.

`g` The genotype of the parent.

`m` The pairing configuration given disomic inheritance. See Gerard et al (2018).

`p` The gamete frequencies. Element `p[[i]]` is the probability a gamete will have dosage `i-1`.

`mode` Whether the inheritance pattern that leads to these gamete frequencies is `"disomic"`, `"polysomic"`, or `"both"`.

Author(s)

David Gerard

References

- Gerard, D., Ferrão, L. F. V., Garcia, A. A. F., & Stephens, M. (2018). Genotyping polyploids from messy sequencing data. Genetics, 210(3), 789-807. doi:10.1534/genetics.118.301468

`seg_lrt` *Test for segregation distortion in a polyploid F1 population.*

Description

Provides tests for segregation distortion for an F1 population of polyploids under various models of meiosis. You can use this test for autopolyploids that exhibit full polysomic inheritance, allopolyploids that exhibit full disomic inheritance, or segmental allopolyploids that exhibit partial preferential pairing. Double reduction is (optionally) fully accounted for in tetraploids, and (optionally) partially accounted for (only at simplex loci) for higher ploidies. Some maximum proportion of outliers can be specified (default at 3%), and so this method can accommodate moderate levels of double reduction at non-simplex loci. Offspring genotypes can either be known, or genotype uncertainty can be represented through genotype likelihoods. Parent data may or may not be provided, at your option. Parents can have different (even) ploidies, at your option. Details of the methods may be found in Gerard et al. (2025).

Usage

```
seg_lrt(
  x,
  p1_ploidy,
  p2_ploidy = p1_ploidy,
  p1 = NULL,
  p2 = NULL,
  model = c("seg", "auto", "auto_dr", "allo", "allo_pp", "auto_allo"),
  outlier = TRUE,
  ret_out = FALSE,
  ob = 0.03,
  db = c("ces", "prcs"),
  ntry = 3,
  opt = c("bobyqa", "L-BFGS-B"),
 optg = c("NLOPT_GN_MLSL_LDS", "NLOPT_GN_ESCH", "NLOPT_GN_CRS2_LM", "NLOPT_GN_ISRES"),
  df_tol = 0.001,
  chisq = FALSE
)
```

Arguments

`x` The data. Can be one of two forms:

- A vector of genotype counts. This is when offspring genotypes are known.
- A matrix of genotype log-likelihoods. This is when there is genotype uncertainty. The rows index the individuals and the columns index the possible genotypes. The genotype log-likelihoods should be base e (natural log).

`p1_ploidy, p2_ploidy` The ploidy of the first or second parent. Should be even.

`p1, p2` One of three forms:

- The known genotype of the first or second parent.
- The vector of genotype log-likelihoods of the first or second parent. Should be base e (natural log).
- `NULL` (completely unknown)

`model` One of six forms:

`"seg"` Segmental allopolyploid. Allows for arbitrary levels of polysomic and disomic inheritance. This can account for partial preferential pairing. It also accounts for double reduction at simplex loci.

`"auto"` Autopolyploid. Allows only for polysomic inheritance. No double reduction.

`"auto_dr"` Autopolyploid, allowing for the effects of double reduction.

`"allo"` Allopolyploid. Only complete disomic inheritance is explored.

`"allo_pp"` Allopolyploid, allowing for the effects of partial preferential pairing. Though, autopolyploid (with complete bivalent pairing and no double reduction) is a special case of this model.

`"auto_allo"` Only complete disomic and complete polysomic inheritance is studied.

`outlier` A logical. Should we allow for outliers (`TRUE`) or not (`FALSE`)?

`ret_out` A logical. Should we return the probability that each individual is an outlier (`TRUE`) or not (`FALSE`)?

`ob` The default upper bound on the outlier proportion.

`db` Should we use the complete equational segregation model (`"ces"`) or the pure random chromatid segregation model (`"prcs"`) to determine the upper bound(s) on the double reduction rate(s). See `drbounds()` for details.

`ntry` The number of times to try the optimization. You probably do not want to touch this.

`opt`	For local optimization, should we use bobyqa (Powell, 2009) or L-BFGS-B (Byrd et al, 1995)? You probably do not want to touch this.
`optg`	Initial global optimization used to start local optimization. Methods are described in the `nloptr` package (Johnson, 2008). You probably do not want to touch this. Possible values are: `"NLOPT_GN_MLSL_LDS"` MLSL (Multi-Level Single-Linkage). Kucherenko and Sytsko (2005); `"NLOPT_GN_ESCH"` ESCH (evolutionary algorithm). da Silva Santos et al. (2010); `"NLOPT_GN_CRS2_LM"` Controlled Random Search (CRS) with local mutation. Kaelo and Ali (2006); `"NLOPT_GN_ISRES"` ISRES (Improved Stochastic Ranking Evolution Strategy). Runarsson and Yao (2005)
`df_tol`	Threshold for the rank of the Jacobian for the degrees of freedom calculation. This accounts for weak identifiability in the null model. You probably do not want to touch this.
`chisq`	A logical. When using known genotypes, this flags to use the chi-squared test or the Likelihood Ratio Test. Default is `FALSE` for the likelihood ratio test.

Value

A list with some or all of the following elements

`stat` The test statistic.

`df` The degrees of freedom of the test.

`p_value` The p-value of the test.

`null_bic` The null model's BIC.

`outprob` Outlier probabilities. Only returned in `ret_out = TRUE`.

- If using genotype counts, element `i` is the probability that an individual *with genotype* `i-1` is an outlier. So the return vector has length ploidy plus 1.
- If using genotype log-likelihoods, element `i` is the probability that individual `i` is an outlier. So the return vector has the same length as the number of individuals.

These outlier probabilities are only valid if the null of no segregation is true.

`null` A list with estimates and information on the null model.

- `l0_pp` Maximized likelihood under the null plus the parent log-likelihoods.
- `l0` Maximized likelihood under using estimated parent genotypes are known parent genotypes.
- `q0` Estimated genotype frequencies under the null.
- `df0` Estimated number of parameters under the null.
- `gam` A list of three lists with estimates of the model parameters. The third list contains the elements `outlier` (which is `TRUE` if outliers were modeled) and `pi` (the estimated outlier proportion). The first two lists contain information on each parent with the following elements:
  - `ploidy` The ploidy of the parent.
  - `g` The (estimated) genotype of the parent.

- `alpha` The estimated double reduction rate(s). `alpha[i]` is the estimated probability that a gamete has `i` copies of identical by double reduction alleles.
- `beta` Double reduction's effect on simplex loci when `type = "mix"` and `add_dr = TRUE`.
- `gamma` The mixing proportions for the pairing configurations. The order is the same as in `seg`.
- `type` Either `"mix"` or `"polysomic"`
- `add_dr` Did we model double reduction at simplex loci when using `type = "mix"` (`TRUE`) or not (`FALSE`)?

`alt` A list with estimates and information on the alternative model.

- `l1` The maximized likelihood under the alternative.
- `q1` The estimated genotype frequencies under the alternative.
- `df1` The estimated number of parameters under the alternative.

Null Model

The gamete frequencies under the null model can be calculated via `gamfreq()`. The genotype frequencies, which are just a discrete linear convolution (`convolve()`) of the gamete frequencies, can be calculated via `gf_freq()`.

The null model's gamete frequencies for true autopolyploids (`model = "auto"`) or true allopolyploids (`model = "allo"`) are given in the `seg` data frame that comes with this package. I only made that data frame go up to ploidy 20, but let me know if you need it for higher ploidies.

The polyRAD folks test for full autopolyploid and full allopolyploid, so I included that as an option (`model = "auto_allo"`).

We can account for arbitrary levels of double reduction in autopolyploids (`model = "auto_dr"`) using the gamete frequencies from Huang et al (2019).

The null model for segmental allopolyploids (`model = "allo_pp"`) is the mixture model of the possible allopolyploid gamete frequencies. The autopolyploid model (without double reduction) is a subset of this mixture model.

In the above mixture model, we can account for double reduction for simplex loci (`model = "seg"`) by just slightly reducing the number of simplex gametes and increasing the number of duplex and nullplex gametes. That is, the frequencies for (nullplex, simplex, duplex) gametes go from `(0.5, 0.5, 0)` to `(0.5 + b, 0.5 - 2 * b, b)`.

`model = "seg"` is the most general, so it is the default. But you should use other models if you have more information on your species. E.g. if you know you have an autopolyploid, use either `model = "auto"` or `model = "auto_dr"`.

Unidentified Parameters

Do NOT interpret the estimated parameters in the `null$gam` list. These parameters are weakly identified (I had to do some fancy spectral methods to account for this in the null distribution of the tests). Even though they are technically identified, you would need a massive data set to be able to estimate them accurately.

Author(s)

David Gerard

References

- Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5), 1190-1208. doi:10.1137/0916069
- da Silva Santos, C. H., Goncalves, M. S., & Hernandez-Figueroa, H. E. (2010). Designing novel photonic devices by bio-inspired computing. *IEEE Photonics Technology Letters*, 22(15), 1177-1179. doi:10.1109/LPT.2010.2051222
- Gerard, D, Ambrosano, GB, Pereira, GdS, & Garcia, AAF (2025). Tests for segregation distortion in higher ploidy F1 populations. *bioRxiv*, p. 1-20. doi:10.1101/2025.06.23.661114
- Huang, K., Wang, T., Dunn, D. W., Zhang, P., Cao, X., Liu, R., & Li, B. (2019). Genotypic frequencies at equilibrium for polysomic inheritance under double-reduction. *G3: Genes, Genomes, Genetics*, 9(5), 1693-1706. doi:10.1534/g3.119.400132
- Johnson S (2008). The NLopt nonlinear-optimization package. `https://github.com/stevengj/nlopt`.
- Kaelo, P., & Ali, M. M. (2006). Some variants of the controlled random search algorithm for global optimization. *Journal of optimization theory and applications*, 130, 253-264. doi:10.1007/s1095700691010
- Kucherenko, S., & Sytsko, Y. (2005). Application of deterministic low-discrepancy sequences in global optimization. *Computational Optimization and Applications*, 30, 297-318. doi:10.1007/s1058900546151
- Powell, M. J. D. (2009), The BOBYQA algorithm for bound constrained optimization without derivatives, Report No. DAMTP 2009/NA06, Centre for Mathematical Sciences, University of Cambridge, UK.
- Runarsson, T. P., & Yao, X. (2005). Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(2), 233-243. doi:10.1109/TSMCC.2004.841906

Examples

```
set.seed(1)
p1_ploidy <- 4
p1 <- 1
p2_ploidy <- 8
p2 <- 4
q <- gf_freq(
  p1_g = p1,
  p1_ploidy = p1_ploidy,
  p1_gamma = 1,
  p1_type = "mix",
  p2_g = p2,
  p2_ploidy = p2_ploidy,
  p2_gamma= c(0.2, 0.2, 0.6),
  p2_type = "mix",
  pi = 0.01)
nvec <- c(stats::rmultinom(n = 1, size = 200, prob = q))
gl <- simgl(nvec = nvec)
seg_lrt(x = nvec, p1_ploidy = p1_ploidy, p2_ploidy = p2_ploidy, p1 = p1, p2 = p2)$p_value
seg_lrt(x = gl, p1_ploidy = p1_ploidy, p2_ploidy = p2_ploidy, p1 = p1, p2 = p2)$p_value
```

seg_multi *Parallelized likelihood ratio test for segregation distortion for arbitrary (even) ploidies.*

Description

Uses the future package to implement parallelization support for the likelihood ratio tests for segregation distortion. Details of this test are provided in the `seg_lrt()` function's documentation. See Gerard et al. (2025) for details of the methods.

Usage

```
seg_multi(
  g,
  p1_ploidy,
  p2_ploidy = p1_ploidy,
  p1 = NULL,
  p2 = NULL,
  model = c("seg", "auto", "auto_dr", "allo", "allo_pp", "auto_allo"),
  outlier = TRUE,
  ret_out = FALSE,
  ob = 0.03,
  db = c("ces", "prcs"),
  ntry = 3,
  df_tol = 0.001
)
```

Arguments

`g` One of two inputs

- A matrix of genotype counts. The rows index the loci and the columns index the genotypes.
- An array of genotype log-likelihoods. The rows index the loci, the columns index the individuals, and the slices index the genotypes. Log-likelihoods are base e (natural log).

`p1_ploidy, p2_ploidy`
The ploidy of the first or second parent. Should be even.

`p1` One of three inputs

- A vector of parent 1's genotypes.
- A matrix of parent 1's genotype log-likelihoods. The rows index the loci and the columns index the genotypes. Logs are in base e (natural log).
- `NULL` (only supported when using genotype likelihoods for the offspring)

`p2`	One of three inputs • A vector of parent 1's genotypes. • A matrix of parent 1's genotype log-likelihoods. The rows index the loci and the columns index the genotypes. Logs are in base e (natural log). • `NULL` (only supported when using genotype likelihoods for the offspring)
`model`	One of six forms: `"seg"` Segmental allopolyploid. Allows for arbitrary levels of polysomic and disomic inheritance. This can account for partial preferential pairing. It also accounts for double reduction at simplex loci. `"auto"` Autopolyploid. Allows only for polysomic inheritance. No double reduction. `"auto_dr"` Autopolyploid, allowing for the effects of double reduction. `"allo"` Allopolyploid. Only complete disomic inheritance is explored. `"allo_pp"` Allopolyploid, allowing for the effects of partial preferential pairing. Though, autopolyploid (with complete bivalent pairing and no double reduction) is a special case of this model. `"auto_allo"` Only complete disomic and complete polysomic inheritance is studied.
`outlier`	A logical. Should we allow for outliers (`TRUE`) or not (`FALSE`)?
`ret_out`	A logical. Should we return the probability that each individual is an outlier (`TRUE`) or not (`FALSE`)?
`ob`	The default upper bound on the outlier proportion.
`db`	Should we use the complete equational segregation model (`"ces"`) or the pure random chromatid segregation model (`"prcs"`) to determine the upper bound(s) on the double reduction rate(s). See `drbounds()` for details.
`ntry`	The number of times to try the optimization. You probably do not want to touch this.
`df_tol`	Threshold for the rank of the Jacobian for the degrees of freedom calculation. This accounts for weak identifiability in the null model. You probably do not want to touch this.

Value

A data frame with the following elements:

`statistic` The likelihood ratio test statistic

`p_value` The p-value of the likelihood ratio test.

`df` The (estimated) degrees of freedom of the test.

`null_bic` The BIC of the null model (no segregation distortion).

`df0` The (estimated) number of parameters under null.

`df1` The (estimated) number of parameters under the alternative.

`p1` The (estimated) genotype of parent 1.

`p2` The (estimated) genotype of parent 2.

`q0` The MLE of the genotype frequencies under the null.

`q1` The MLE of the genotype frequencies under the alternative.

`outprob` Outlier probabilities. Only returned in `ret_out = TRUE`.

- If using genotype counts, element `i` is the probability that an individual *with genotype* `i-1` is an outlier. So the return vector has length ploidy plus 1.
- If using genotype log-likelihoods, element `i` is the probability that individual `i` is an outlier. So the return vector has the same length as the number of individuals.

These outlier probabilities are only valid if the null of no segregation is true.

Note that since this data frame contains the list-columns `q0` and `q1`, you cannot use `write.csv()` to save it. You have to either remove those columns first or use something like `saveRDS()`

Parallel Computation

The `seg_multi()` function supports parallel computing. It does so through the future package.

You first specify the evaluation plan with `plan()` from the `future` package. On a local machine, this is typically just `future::plan(future::multisession, workers = nc)` where `nc` is the number of workers you want. You can find the maximum number of possible workers with `availableCores()`. You then run `seg_multi()`, then shut down the workers with `future::plan(future::sequential)`. The pseudo code is

```
future::plan(future::multisession, workers = nc)
seg_multi()
future::plan(future::sequential)
```

Null Model

The gamete frequencies under the null model can be calculated via `gamfreq()`. The genotype frequencies, which are just a discrete linear convolution (`convolve()`) of the gamete frequencies, can be calculated via `gf_freq()`.

The null model's gamete frequencies for true autopolyploids (`model = "auto"`) or true allopolyploids (`model = "allo"`) are given in the `seg` data frame that comes with this package. I only made that data frame go up to ploidy 20, but let me know if you need it for higher ploidies.

The polyRAD folks test for full autopolyploid and full allopolyploid, so I included that as an option (`model = "auto_allo"`).

We can account for arbitrary levels of double reduction in autopolyploids (`model = "auto_dr"`) using the gamete frequencies from Huang et al (2019).

The null model for segmental allopolyploids (`model = "allo_pp"`) is the mixture model of the possible allopolyploid gamete frequencies. The autopolyploid model (without double reduction) is a subset of this mixture model.

In the above mixture model, we can account for double reduction for simplex loci (`model = "seg"`) by just slightly reducing the number of simplex gametes and increasing the number of duplex and nullplex gametes. That is, the frequencies for (nullplex, simplex, duplex) gametes go from `(0.5, 0.5, 0)` to `(0.5 + b, 0.5 - 2 * b, b)`.

`model = "seg"` is the most general, so it is the default. But you should use other models if you have more information on your species. E.g. if you know you have an autopolyploid, use either `model = "auto"` or `model = "auto_dr"`.

Unidentified Parameters

Do NOT interpret the estimated parameters in the `null$gam` list. These parameters are weakly identified (I had to do some fancy spectral methods to account for this in the null distribution of the tests). Even though they are technically identified, you would need a massive data set to be able to estimate them accurately.

Author(s)

David Gerard

References

- Gerard, D, Ambrosano, GB, Pereira, GdS, & Garcia, AAF (2025). Tests for segregation distortion in higher ploidy F1 populations. *bioRxiv*, p. 1-20. doi:10.1101/2025.06.23.661114

See Also

- `seg_lrt()` Single locus LRT for segregation distortion.
- `gamfreq()` Gamete frequencies under various models of meiosis
- `gf_freq()` F1 genotype frequencies under various models of meiosis.
- `multidog_to_g()` Converts the output of `updog::multidog()` into something that you can input into `seg_multi()`.

Examples

```
## Assuming genotypes are known (typically a bad idea)
glist <- multidog_to_g(
  mout = ufit,
  ploidy = 4,
  type = "all_g",
  p1 = "indigocrisp",
  p2 = "sweetcrisp")
p1_1 <- glist$p1
p2_1 <- glist$p2
g_1 <- glist$g
s1 <- seg_multi(
  g = g_1,
  p1_ploidy = 4,
  p2_ploidy = 4,
  p1 = p1_1,
  p2 = p2_1)
s1[, c("snp", "p_value")]

## Put NULL if you have absolutely no information on the parents
s2 <- seg_multi(g = g_1, p1_ploidy = 4, p2_ploidy = 4, p1 = NULL, p2 = NULL)
s2[, c("snp", "p_value")]

## Using genotype likelihoods (typically a good idea)
## Also demonstrate parallelization through future package.
glist <- multidog_to_g(
  mout = ufit,
  ploidy = 4,
  type = "all_gl",
  p1 = "indigocrisp",
  p2 = "sweetcrisp")
p1_2 <- glist$p1
p2_2 <- glist$p2
g_2 <- glist$g

# future::plan(future::multisession, workers = 2)
# s3 <- seg_multi(
#   g = g_2,
#   p1_ploidy = 4,
#   p2_ploidy = 4,
#   p1 = p1_2,
#   p2 = p2_2,
#   ret_out = TRUE)
# future::plan(future::sequential)
# s3[, c("snp", "p_value")]

## Outlier probabilities are returned if `ret_out = TRUE`
# graphics::plot(s3$outprob[[6]], ylim = c(0, 1))
```

---

simf1g *Simulate genotype counts from F1 individuals*

Description

Simulate genotype counts from F1 individuals

Usage

```
simf1g(n, g1, g2, alpha = 0, xi1 = 1/3, xi2 = 1/3)
```

Arguments

n	Sample size.
g1	The first parent's genotype.
g2	The second parent's genotype.
alpha	The double reduction rate.
xi1	The first parent's preferential pairing parameter.
xi2	The second parent's preferential pairing parameter.

Value

A vector of counts, where element `i` is the number of simulated individuals with genotype `i-1`.

Author(s)

David Gerard

Examples

```
set.seed(1)
simf1g(n = 10, g1 = 1, g2 = 2)
```

---

simf1gl *Simulate genotype likelihoods of F1 individuals.*

Description

Simulate genotype likelihoods of F1 individuals.

Usage

```
simf1gl(n, g1, g2, rd = 10, alpha = 0, xi1 = 1/3, xi2 = 1/3)
```

Arguments

`n`	Sample size.
`g1`	The first parent's genotype.
`g2`	The second parent's genotype.
`rd`	The read depth.
`alpha`	The double reduction rate.
`xi1`	The first parent's preferential pairing parameter.
`xi2`	The second parent's preferential pairing parameter.

Value

The matrix of offspring genotype log-likelihoods.

Author(s)

David Gerard

Examples

```
set.seed(1)
simf1gl(n = 10, g1 = 1, g2 = 2)
```

simgl — *Simulate genotype (log) likelihoods from genotype counts*

Description

Provide a vector of genotype counts and this will return a matrix of genotype log-likelihoods.

Usage

```
simgl(nvec, rd = 10, seq = 0.01, bias = 1, od = 0.01)
```

Arguments

`nvec`	A vector of counts. `nvec[k]` is the number of folks with a genotype of k-1.
`rd`	Read depth. Lower is more uncertain.
`seq`	Sequencing error rate. Higher means more uncertain.
`bias`	Bias. 1 means no bias.
`od`	Overdispersion. Typical value is like 0.01. Higher means more uncertain.

Value

A matrix of genotype log-likelihoods. The rows index the individuals and the columns index the genotypes. This is natural log (base e).

Author(s)

David Gerard

Examples

```
set.seed(1)
simgl(nvec = c(1, 2, 1, 1, 3))
```

three_to_two — *Convert from three parameters to two parameters*

Description

Convert from three parameters to two parameters

Usage

```
three_to_two(tau, beta, gamma)
```

Arguments

`tau`	Probability of quadrivalent formation
`beta`	Probability of double reduction given quadrivalent formation
`gamma`	Probability of AA/aa pairing given bivalent formation

Value

A vector of length two. The first is the double reduction rate (`alpha`), and the second is the preferential pairing parameter (`xi`).

Author(s)

David Gerard

Examples

```
three_to_two(tau = 0.1, beta = 1/6, gamma = 1/4)
```

`ufit` *Genotype data from Cappai et al. (2020)*

Description

A subset of data from Cappai et al. (2020), fit using `multidog()`. This just contains a random set of 10 loci.

Usage

```
ufit

ufit2

ufit3
```

Format

An object of type `multidog` output from `multidog()`.

`ufit` Uses the `model = "norm"` option.

`ufit2` Uses the `model = "f1pp"` option.

`ufit3` Uses the `model = "f1"` option.

An object of class `multidog` of length 2.

An object of class `multidog` of length 2.

Source

doi:10.5281/zenodo.13715703

References

- Cappai, F., Amadeu, R. R., Benevenuto, J., Cullen, R., Garcia, A., Grossman, A., Ferrão, L., & Munoz, P. (2020). High-resolution linkage map and QTL analyses of fruit firmness in autotetraploid blueberry. *Frontiers in plant science*, 11, 562171. doi:10.3389/fpls.2020.562171.

Index