AlphaMate (Version 0.1.0)

Contents

Introduction

AlphaMate is a program for optimising selection, maintenance of diversity, and mate allocation in breeding programs. This optimisation finds which individuals should contribute to the next generation and potentially how should these individuals be mated / crossed to deliver desired targets. **AlphaMate** works with animal and self- and cross-pollinating plant populations.

Please report bugs or suggestions on how the program / user interface / manual could be improved or made more user friendly to Gregor Gorjanc (Gregor.Gorjanc@roslin.ed.ac.uk) or John Hickey (John.Hickey@roslin.ed.ac.uk).

Availability

AlphaMate is available from AlphaGenes website (http://www.AlphaGenes.roslin.ed.ac.uk/AlphaMate). Material available comprises the compiled programs for Windows, Linux, and Mac OSX platforms, this document, and examples.

Conditions of use

AlphaMate is available to the scientific community free of charge. Users are required, however, to credit its use in any publications. Commercial users should contact Gregor Gorjanc (Gregor.Gorjanc@roslin.ed.ac.uk) or John Hickey (John.Hickey@roslin.ed.ac.uk).

Suggested Citation:

Gorjanc, G, Hickey, JM (2018). *AlphaMate: a program for optimising selection, maintenance of diversity, and mate allocation in breeding programs*. bioRxiv 250837; doi: http://doi.org/10.1101/10.1101/250837.

Disclaimer

While every effort has been made to ensure that **AlphaMate** does what it claims to do, there is absolutely no guarantee that the results provided are correct. Use of **AlphaMate** is entirely at your own risk!

Advertisement

AlphaMate is part of the AlphaSuite collection of programs that we have developed. The collection can perform many of the common tasks in animal breeding, plant breeding, and human genetics including phasing, imputation, estimation of genetic values, genome-wide association, optimal contributions, simulation, and various data recoding and handling tools.

The AlphaSuite is available from AlphaGenes website (http://www.AlphaGenes.roslin.ed.ac.uk).

General overview

AlphaMate optimises selection, maintenance of diversity, and mate allocation in breeding programs. It achieves this by jointly optimising contributions and mate allocations. Mate allocation is optional, but turned on by default. The goal of this optimisation is to find a **valid mating plan** (i.e., which individuals contribute and how they are mated / crossed) that delivers **desired targets**. This is achieved with an evolutionary optimisation of a single objective or multiple objectives simultaneously.

A **valid mating plan** is defined by a combination of mating constraints (see Mating constraints), which cover reproductive systems in animals and cross- and self-pollinating plants:

i) the number of matings,

ii) the maximal number of parents,

iii) the minimal, equal, or maximal number of contributions per parent, or

iv) allowance for selfing.

The **desired targets** (see Desired targets) formulate optimisation objectives, such as:

a) maximize Genetic gain,

b) minimize the Loss of genetic diversity,

c) minimize Inbreeding depression,

d) maximize Genetic gain with the constrained Loss of genetic diversity or Inbreeding depression, or

e) as a) or d) but with the ability to genome edit a fixed set of contributors.

Optimisation is performed with an evolutionary algorithm based on differential evolution (Storn and Price, 1997) with modifications to avoid premature convergence (Gondro and Kinghorn, 2009). For a single target (a-c optimisation objectives), **AlphaMate** optimises a single objective function accounting for matings constraints. For multiple targets, **AlphaMate** performs multiple objective optimisation in two steps; see e.g. Deb (2014) for review. First, **AlphaMate** optimises single objective functions for each target separately to find bounds of the objective space and normalize objectives. Second, **AlphaMate** uses the ε-constraint method to either: i) find a Pareto-optimal solution with targeted balance between objectives or ii) evaluate the whole frontier of Pareto-optimal solutions (the Pareto frontier). A Pareto-optimal solution is the best solution with a specific balance between objectives. The Pareto frontier is a set of Pareto-optimal solutions and is useful when a breeder does not have clearly defined targets and can explore optimal solutions with different balance between targets to reach a decision. Figure 1 below demonstrates the Pareto frontier of genetic gain and coancestry and the optimisation path for a targeted solution. Further details about implemented objective functions are provided in description of the specifications file (see Optimisation controls).

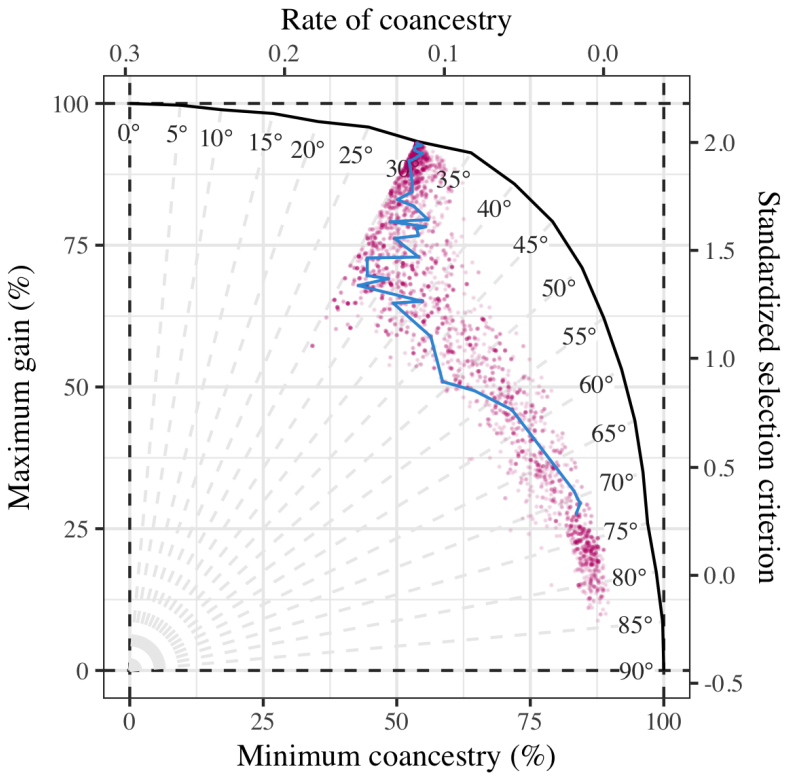

Figure 1: Trade-off between genetic gain and coancestry and optimisation path of evolutionary algorithm (target set to 30°, dots show evaluated solutions, line shows evolution of the best solution)

Optimisation works with mating plans, which **AlphaMate** encodes as proposed by Kinghorn and Shepherd (1999). **AlphaMate** ensures that mating plans are valid in two ways. First, **AlphaMate** fixes encoded representation, e.g., it trims contributions to user defined limits and round them to integer values (Lampinen and Zelinka, 1999). Second, when fixing is not sufficient, **AlphaMate** penalizes invalid mating plan so that evolutionary algorithm advances (more) valid mating plans.

A note on the reproducibility of results. **AlphaMate** uses an evolutionary algorithm, which can give different solutions in repeated runs. We find however, that the result summaries do not vary a lot, provided that convergence has been achieved (see Optimisation controls). To ensure reproducibility users can fix the seed value for random number generator via SeedFile or Seed specifications.

A note about the speed and scaling of **AlphaMate** with large populations. **AlphaMate** finds good solutions quickly for applications with up to a couple of thousands of individuals. When the number of individuals is larger than this, **AlphaMate** has to evaluate a very large number of combinations and optimisation takes a long time. There are several options to speed-up this optimisation. One option is to pre-select individuals, which you can do

yourself or use an **AlphaMate** specification that accounts for this pre-selection in further optimisation, albeit at a higher computational cost. Another option is to tweak optimisation controls. Work on a scalable solution is in progress.

Run command

AlphaMate is run from a terminal. By default it uses a Specifications file named `AlphaMateSpec.txt`, but users can provide other specifications file.

On MS Windows use one of the following options:

- `AlphaMate.exe`
- `AlphaMate.exe AnotherSpecFile.txt`

On Linux or Mac OSX use one of the following options:

- `./AlphaMate`
- `./AlphaMate AnotherSpecFile.txt`

Specifications file

AlphaMate is controlled by a single specification file, which should be a simple text file. In this file, a user specifies:

The specifications are of the form:

```
Specification , Value
```

The specifications can appear in any order. In the case of conflicting specifications (e.g., one turns a feature off, but another turns it on), the latest one is used.

Specifications are not case sensitive.

Unrecognised specifications are ignored, so beware of typos! User is notified which specifications are accepted and which are ignored.

An example:

```
GenderFile            , Gender.txt
CoancestryMatrixFile  , Coancestry.txt
SelCriterionFile      , Criterion.txt
NumberOfMatings       , 500
NumberOfMaleParents   ,  25
NumberOfFemaleParents , 500
ModeMinCoancestry     , Yes
Stop
```

There are further examples provided in the Examples.

Input Files

The basic input files are coancestry or numerator relationship matrix, selection criteria, and gender information for candidates. Coancestry matrix and selection criteria can be based on pedigree or genomic data, whichever the user provides. Additionally, further individual or mating specific information can be given to enrich optimisation objectives.

CoancestryMatrixFile

```
CoancestryMatrixFile , [String]
CoancestryMatrixFile , Coancestry.txt
```

CoancestryMatrixFile specifies a file that contains Coancestry values between individuals. The file should have *n + 1* columns and *n* rows for *n* individuals. The first column contains individual identifications. An example for a family with two parents and two offspring is:

```
id1 0.50 0.00 0.25 0.25
id2 0.00 0.50 0.25 0.25
id3 0.25 0.25 0.50 0.25
id4 0.25 0.25 0.25 0.50
```

This specification can be used instead of NrmMatrixFile.

NrmMatrixFile

```
NrmMatrixFile , [String]
NrmMatrixFile , Nrm.txt
```

NrmMatrixFile specifies a file that contains Numerator relationship values between individuals. The file should have *n + 1* columns and *n* rows for *n* individuals. The first column contains individual identifications. An example for a family with two parents and two offspring is:

```
id1 1.0 0.0 0.5 0.5
id2 0.0 1.0 0.5 0.5
id3 0.5 0.5 1.0 0.5
id4 0.5 0.5 0.5 1.0
```

This specification can be used instead of CoancestryMatrixFile.

SelCriterionFile

```
SelCriterionFile , [String]
SelCriterionFile , Criterion.txt
```

SelCriterionFile specifies a file that contains Selection criterion. The file should have *2* columns and *n* rows for *n* individuals. The first column contains individual identifications. An example is:

```
id1 15.87
id2 16.84
id3 16.11
...
```

In case genome editing functionality is used, the file should be expanded with an additional column, that gives selection criterion after genome editing. An example is:

```
id1 15.87 16.00
id2 16.84 16.95
id3 16.11 16.22
...
```

GenderFile

```
GenderFile , [String]
GenderFile , Gender.txt
```

GenderFile specifies a file that contains gender information. The file should have *2* columns and *n* rows for *n* individuals. The first column contains individual identifications. The second column contains gender codes. The codes must be either 1 or 2. An example is:

```
id1 1
id2 2
id3 1
...
```

Tip: Note that the "gender" functionality can also be used in cases where biological gender is not relevant. For example in some plant populations we might have two pools of individuals and want to make cross between these two pools.

GenericIndividualCriterionFile

This is an experimental specification in development and no yet tested! Do not use!

```
GenericIndividualCriterionFile , [String]
GenericIndividualCriterionFile , AddCriteriaPerIndividual.txt
```

GenericIndividualCriterionFile specifies a file that contains additional individual specific criteria to enrich optimisation. These criteria are included in the objective function by GenericMatingCriterionWeight. The file should have *k + 1* columns and *n* rows for *n* individuals. The first column contains individual identifications. An example is:

```
id1  1.0 22.0
id2  0.5 12.0
id3 -0.1  5.0
...
```

GenericIndividualCriterionColumns

This is an experimental specification in development and no yet tested! Do not use!

```
GenericIndividualCriterionColumns , [Value]
GenericIndividualCriterionColumns , 2
```

GenericIndividualCriterionColumns specifies how many columns from the GenericIndividualCriterionFile should be used.

GenericMatingCriterionFile

This is an experimental specification in development and no yet tested! Do not use!

```
GenericMatingCriterionFile , [String]
GenericMatingCriterionFile , AddCriteriaPerMating.txt
```

GenericMatingCriterionFile specifies a file that contains additional mating specific criteria to enrich optimisation. These criteria are included in the objective function by GenericMatingCriterionWeight. The file should have *k + 2* columns and *n * n* rows for *n* individuals. The first / second column contains individual identifications of the first / second individual of a mating / cross. An example is:

```
id1 id1  0.0  0.01
id1 id2 -0.1  0.10
id1 id3  0.5 -0.02
...
```

GenericMatingCriterionColumns

This is an experimental specification in development and no yet tested! Do not use!

```
GenericMatingCriterionColumns , [Value]
GenericMatingCriterionColumns , 2
```

GenericMatingCriterionColumns specifies how many columns from the GenericMatingCriterionFile should be used.

Mating constraints

Mating constraints can be gender specific or generic to accommodate different reproductive systems in animals and cross- or self-pollinating plants (see GenderFile). A user can specify all the mating constraints or a subset of them depending on the objective of optimisation and biologic or logistic reasons.

NumberOfMatings

```
NumberOfMatings , [Value]
NumberOfMatings , 100
```

NumberOfMatings specifies the number of matings that should be performed.

NumberOfParents

```
NumberOfParents , [Value]
NumberOfParents , 10
```

NumberOfParents specifies the number of parents that should be maximally used. This specification is used when "gender" functionality is NOT used (see GenderFile).

NumberOfMaleParents

```
NumberOfMaleParents , [Value]
NumberOfMaleParents , 10
```

NumberOfMaleParents specifies the number of male parents that should be maximally used. This specification is used when "gender" functionality is used (see GenderFile).

NumberOfFemaleParents

```
NumberOfFemaleParents , [Value]
NumberOfFemaleParents , 10
```

NumberOfFemaleParents specifies the number of female parents that should be maximally used. This specification is used when "gender" functionality is used (see GenderFile).

EqualizeContributions

```
EqualizeContributions , [Yes/No]
EqualizeContributions , Yes
```

EqualizeContributions specifies whether parent Contributions should be equal, i.e., all chosen parents contribute the same number of matings. Default is `No`. This specification is used when "gender" functionality is NOT used (see GenderFile).

EqualizeMaleContributions

```
EqualizeMaleContributions , [Yes/No]
EqualizeMaleContributions , Yes
```

EqualizeMaleContributions specifies whether male parent Contributions should be equal, i.e., all chosen male parents contribute the same number of matings. Default is `No`. This specification is used when "gender" functionality is used (see GenderFile).

EqualizeFemaleContributions

```
EqualizeFemaleContributions , [Yes/No]
EqualizeFemaleContributions , Yes
```

EqualizeFemaleContributions specifies whether female parent Contributions should be equal, i.e., all chosen female parents contribute the same number of matings. Default is `No`. This specification is used when "gender" functionality is used (see GenderFile).

LimitContributions

```
LimitContributions , [Yes/No]
LimitContributions , Yes
```

LimitContributions specifies whether Contributions should be limited with LimitContributionsMin and LimitContributionsMax. Default is `No`. This specification is used when "gender" functionality is NOT used (see GenderFile).

LimitContributionsMin

```
LimitContributionsMin , [Value]
LimitContributionsMin , 5
```

LimitContributionsMin specifies minimum allowed Contributions per parent. This specification is used when LimitContributions is active and "gender" functionality is NOT used (see GenderFile). When the limit cannot be accommodated, the objective function is penalized by LimitContributionsMinWeight.

LimitContributionsMax

```
LimitContributionsMax , [Value]
LimitContributionsMax , 5
```

LimitContributionsMax specifies maximum allowed Contributions per parent. This specification is used when LimitContributions is active and "gender" functionality is NOT used (see GenderFile).

LimitMaleContributions

```
LimitMaleContributions , [Yes/No]
LimitMaleContributions , Yes
```

LimitMaleContributions specifies whether male parent Contributions should be limited with LimitMaleContributionsMin and LimitMaleContributionsMax. Default is `No`. This specification is used when "gender" functionality is used (see GenderFile).

LimitMaleContributionsMin

```
LimitMaleContributionsMin , [Value]
LimitMaleContributionsMin , 5
```

LimitMaleContributionsMin specifies minimum allowed Contributions per male parent. This specification is used when LimitMaleContributions is active and "gender" functionality is used (see GenderFile). When the limit cannot be accommodated, the objective function is penalized by LimitMaleContributionsMinWeight.

LimitMaleContributionsMax

```
LimitMaleContributionsMax , [Value]
LimitMaleContributionsMax , 5
```

LimitMaleContributionsMax specifies maximum allowed Contributions per male parent. This specification is used when LimitMaleContributions is active and "gender" functionality is used (see GenderFile).

LimitFemaleContributions

```
LimitFemaleContributions , [Yes/No]
LimitFemaleContributions , Yes
```

LimitFemaleContributions specifies whether female parent Contributions should be limited with LimitFemale-ContributionsMin and LimitFemaleContributionsMax. Default is `No`. This specification is used when "gender" functionality is used (see GenderFile).

LimitFemaleContributionsMin

```
LimitFemaleContributionsMin , [Value]
LimitFemaleContributionsMin , 5
```

LimitFemaleContributionsMin specifies minimum allowed Contributions per female parent. This specification is used when LimitFemaleContributions is active and "gender" functionality is used (see GenderFile). When the limit cannot be accommodated, the objective function is penalized by LimitFemaleContributionsMinWeight.

LimitFemaleContributionsMax

```
LimitFemaleContributionsMax , [Value]
LimitFemaleContributionsMax , 5
```

LimitFemaleContributionsMax specifies maximum allowed Contributions per male parent. This specification is used when LimitFemaleContributions is active and "gender" functionality is used (see GenderFile).

AllowSelfing

```
AllowSelfing , [Yes/No]
AllowSelfing , Yes
```

AllowSelfing specifies whether selfing is allowed. Default is `No`. This specification is only relevant when "gender" functionality is NOT used (see GenderFile) and when mate allocation is used (MateAllocation).

Desired targets

Desired targets quantify breeders' direction and define the optimisation objectives (see Optimisation controls). For ease of use **AlphaMate** provides different ways to specify targets, e.g., constraint on the Loss of genetic diversity can be defined with the targeted value of:

a) Group coancestry,

b) Rate of coancestry,

c) Minimum percentage coancestry, or

d) Trigonometric degrees between genetic gain and group coancestry (see Figure 1).

Similar flexibility of defining targets is available for selection and inbreeding.

Some targets are conflicting (selection and maintenance of diversity), while others are less so (loss of diversity and inbreeding). To this end, a user can specify several targets for selection and maintenance of diversity to explore the space of possibilities within one **AlphaMate** run, while only one specification for inbreeding is allowed within one **AlphaMate** run.

The most "practical" target specifications that work in many different settings are: TargetMaxCriterionPct, TargetMinCoancestryPct, TargetDegree, and TargetMinInbreedingPct.

A specification file part with targets might therefore look like this (if this was desired):

```
TargetMaxCriterionPct  , 99
TargetMaxCriterionPct  , 95
TargetMaxCriterionPct  , 90
TargetMaxCriterionPct  , 85
...
TargetMinCoancestryPct , 17
TargetMinCoancestryPct , 25
...
TargetMinInbreedingPct , 50
...
```

TargetSelCriterion

```
TargetSelCriterion , [Value]
TargetSelCriterion , 100
```

TargetSelCriterion specifies the value for Selection criterion that **AlphaMate** should target. Note that this value depends on the scale of provided values! TargetSelCriterionStd is a scale free alternative, while TargetMaxCriterionPct is a scale free and normalized alternative.

TargetSelCriterionStd

```
TargetSelCriterionStd , [-5, 5]
TargetSelCriterionStd , 2
```

TargetSelCriterionStd specifies the value for standardized Selection criterion that **AlphaMate** should target. TargetMaxCriterionPct is a normalized alternative.

TargetMaxCriterionPct

```
TargetMaxCriterionPct , [0, 100]
TargetMaxCriterionPct , 90
```

TargetMaxCriterionPct specifies the value for Maximum percentage gain that **AlphaMate** should target.

TargetCoancestry

```
TargetCoancestry , [-1, +1]
TargetCoancestry , 0.35
```

TargetCoancestry specifies the value for Group coancestry that **AlphaMate** should target. Note that this value depends on the scale of provided Coancestry values! TargetCoancestryRate is a scale free alternative, while TargetMinCoancestryPct is a normalized alternative.

TargetCoancestryRate

```
TargetCoancestryRate , [-1, +1]
TargetCoancestryRate , 0.05
```

TargetCoancestryRate specifies the value for the Rate of coancestry that **AlphaMate** should target. TargetMinCoancestryPct is a normalized alternative.

TargetMinCoancestryPct

```
TargetMinCoancestryPct , [0, 100]
TargetMinCoancestryPct , 75
```

TargetMinCoancestryPct specifies the value for Minimum percentage coancestry that **AlphaMate** should target.

TargetDegree

```
TargetDegree , [0, 90]
TargetDegree , 45
```

TargetDegree specifies the value for Trigonometric degrees that **AlphaMate** should target.

EvaluateFrontier

```
EvaluateFrontier , [Yes/No]
EvaluateFrontier , Yes
```

EvaluateFrontier specifies whether the whole Pareto frontier of genetic gain and group coancestry should be evaluated. Default is `No`. When actived, **AlphaMate** runs optimisations with Trigonometric degrees of 10, 20, ..., 90.

TargetInbreeding

```
TargetInbreeding , [-1, +1]
TargetInbreeding , 0.15
```

TargetInbreeding specifies the value for Inbreeding that **AlphaMate** should target. Note that this value depends on the scale of provided Coancestry values! TargetInbreedingRate is a scale free alternative, while TargetMinInbreedingPct is a normalized alternative.

TargetInbreedingRate

```
TargetInbreedingRate , [-1, +1]
TargetInbreedingRate , 0.025
```

TargetInbreedingRate specifies the value for the Rate of inbreeding that **AlphaMate** should target. TargetMinInbreedingPct is a normalized alternative.

TargetMinInbreedingPct

```
TargetMinInbreedingPct , [0, 100]
TargetMinInbreedingPct , 75
```

TargetMinInbreedingPct specifies the value for Minimum percentage coancestry that **AlphaMate** should target.

Optimisation controls

Optimisation controls specify optimisation objectives (modes), penalties used to penalize invalid mating plans, weights used to combine multiple targets into an objective function, and parameters of evolutionary algorithm such as the number of iterations, the number of evaluated matings plans per iteration, convergence criteria, etc.

Note that you do not need to specify the optimisation objectives (modes) directly. This is done automatically by defining desired targets (Desired targets). However, should you wish to, you can switch different optimisations on directly.

ModeMinCoancestry

```
ModeMinCoancestry , [Yes/No]
ModeMinCoancestry , Yes
```

ModeMinCoancestry minimizes the Rate of coancestry. The objective function for this optimisation is $f = -\triangle C$, where $\triangle C$ is the Rate of coancestry of an evaluated solution.

ModeMinInbreeding

```
ModeMinInbreeding , [Yes/No]
ModeMinInbreeding , Yes
```

ModeMinInbreeding minimizes the Rate of inbreeding. The objective function for this optimisation is $f = -\triangle F$, where $\triangle F$ is the Rate of inbreeding of an evaluated solution.

ModeMaxCriterion

```
ModeMaxCriterion , [Yes/No]
ModeMaxCriterion , Yes
```

ModeMaxCriterion maximizes standardized Genetic gain. The objective function for this optimisation is $f = \triangle \bar{z}$, where $\triangle \bar{z}$ is the standardized Genetic gain of an evaluated solution.

ModeOpt

```
ModeOpt , [Yes/No]
ModeOpt , Yes
```

ModeOpt maximizes standardized Genetic gain with constraint on Group coancestry and potentially other targets (e.g., Inbreeding). The objective function for this optimisation is $f = \triangle \bar{z}_n / \triangle \bar{z}_{n,t}$, where $\triangle \bar{z}_n$ is the Maximum percentage gain of an evaluated solution and $\triangle \bar{z}_{n,t}$ is the targeted Maximum percentage gain. Constraint on Group coancestry is achieved by adding the following term to the objective function, but only when Trigonometric degrees are above the targeted value: $w \times |1 - d/d_t|$, where w is CoancestryWeight, d are Trigonometric degrees of an evaluated solution, and d_t are targeted Trigonometric degrees. Similar term is added for Inbreeding via InbreedingWeight.

MateAllocation

```
MateAllocation , [Yes/No]
MateAllocation , Yes
```

MateAllocation activates mate allocation. Default is *Yes*.

RandomMateAllocation

This is an experimental specification in development and no yet tested! Do not use!

```
RandomMateAllocation , [Yes/No]
RandomMateAllocation , Yes
```

RandomMateAllocation activates random mate allocation. Default is *No*.

Preselect

```
Preselect , [Yes/No]
Preselect , Yes
```

Preselect activates preselection of individuals prior to running the ModeMaxCriterion and ModeOpt. Default is *No*.

PreselectPercentage

```
PreselectPercentage , [0, 100]
PreselectPercentage , 50
```

PreselectPercentage specifies percentage of individuals preselected when running the ModeMaxCriterion and ModeOpt. This specification is used when Preselect is active and "gender" functionality is NOT used (see GenderFile).

PreselectMales

```
PreselectMales , [Yes/No]
PreselectMales , Yes
```

PreselectMales activates preselection of males prior to running the ModeMaxCriterion and ModeOpt. Default is *No*.

PreselectMalesPercentage

```
PreselectMalesPercentage , [0, 100]
PreselectMalesPercentage , 50
```

PreselectMalesPercentage specifies percentage of males preselected when running the ModeMaxCriterion and ModeOpt. This specification is used when PreselectMales is active and "gender" functionality is used (see GenderFile).

PreselectFemales

```
PreselectFemales , [Yes/No]
PreselectFemales , Yes
```

PreselectFemales activates preselection of females prior to running the ModeMaxCriterion and ModeOpt. Default is *No*.

PreselectFemalesPercentage

```
PreselectFemalesPercentage , [0, 100]
PreselectFemalesPercentage , 50
```

PreselectFemalesPercentage specifies percentage of females preselected when running the ModeMaxCriterion and ModeOpt. This specification is used when PreselectMales is active and "gender" functionality is used (see GenderFile).

PAGE

```
PAGE , [Yes/No]
PAGE , Yes
```

PAGE activates genome editing (promotion of alleles by genome editing). Default is *No*.

PAGEMax

```
PAGEMax , [Value]
PAGEMax , 5
```

PAGEMax specifies how many individuals can be genome edited. This specification is used when PAGE is active and "gender" functionality is NOT used (see GenderFile).

PAGEMales

```
PAGEMales , [Yes/No]
PAGEMales , Yes
```

PAGEMales activates genome editing of males (promotion of alleles by genome editing). Default is *No*.

PAGEMalesMax

```
PAGEMalesMax , [Value]
PAGEMalesMax , 5
```

PAGEMalesMax specifies how many males can be genome edited. This specification is used when PAGEMales is active and "gender" functionality is used (see GenderFile).

PAGEFemales

```
PAGEFemales , [Yes/No]
PAGEFemales , Yes
```

PAGEFemales activates genome editing of females (promotion of alleles by genome editing). Default is *No*.

PAGEFemalesMax

```
PAGEFemalesMax , [Value]
PAGEFemalesMax , 5
```

PAGEFemalesMax specifies how many females can be genome edited. This specification is used when PAGEFemales is active and "gender" functionality is used (see GenderFile).

CoancestryWeight

```
CoancestryWeight , [Value]
CoancestryWeight , -2.0
```

CoancestryWeight specifies the weight that is placed on deviation from the targeted "level of coancestry". Default is *-1.0*. Note that positive weight encourages higher coancestry.

CoancestryWeightBellow

```
CoancestryWeightBellow , [Yes/No]
CoancestryWeightBellow , Yes
```

CoancestryWeightBellow specifies whether **AlphaMate** should penalize solutions that have lower than targeted/constrained "level of coancestry". Default is *No*.

InbreedingWeight

```
InbreedingWeight , [Value]
InbreedingWeight , -2.0
```

InbreedingWeight specifies the weight that is placed on deviation from the targeted "level of inbreeding". Default is *-1.0*. Note that positive weight encourages higher inbreeding.

LimitContributionsMinWeight

```
LimitContributionsMinWeight , [Value]
LimitContributionsMinWeight , -2.0
```

LimitContributionsMinWeight specifies the weight to penalize solutions that could not limit the minimum number of Contributions for parents (LimitContributionsMin). **AlphaMate** tries to avoid this by fixing solutions, but sometimes fixing is not effective. The value of LimitContributionsMinWeight * Diff is added to objective function to penalize every breaking instance, where Diff is the difference between the achieved and limited Contributions. The default value is -1.0. Note that positive values encourage breaking the limit.

LimitMaleContributionsMinWeight

```
LimitMaleContributionsMinWeight , [Value]
LimitMaleContributionsMinWeight , -2.0
```

LimitMaleContributionsMinWeight specifies the weight to penalize solutions that could not limit the minimum number of Contributions for male parents (LimitMaleContributionsMin). **AlphaMate** tries to avoid this by fixing solutions, but sometimes fixing is not effective. The value of LimitMaleContributionsMinWeight * Diff is added to objective function to penalize every breaking instance, where Diff is the difference between the achieved and limited Contributions. The default value is -1.0. Note that positive values encourage breaking the limit.

LimitFemaleContributionsMinWeight

```
LimitFemaleContributionsMinWeight , [Value]
LimitFemaleContributionsMinWeight , -2.0
```

LimitFemaleContributionsMinWeight specifies the weight to penalize solutions that could not limit the minimum number of Contributions for female parents (LimitFemaleContributionsMin). **AlphaMate** tries to avoid this by fixing solutions, but sometimes fixing is not effective. The value of LimitFemaleContributionsMinWeight * Diff is added to objective function to penalize every breaking instance, where Diff is the difference between the achieved and limited Contributions. The default value is -1.0. Note that positive values encourage breaking the limit.

SelfingWeight

```
SelfingWeight , [Value]
SelfingWeight , -2.0
```

SelfingWeight specifies the weight to penalize solutions that involve some degree of selfing, when selfing should not happen, i.e., when AllowSelfing is `No` and "gender" functionality is NOT used. **AlphaMate** tries to avoid this by fixing solutions, but sometimes fixing is not effective. The value of SelfingWeight is added to objective function to penalize every selfing instance. The default value is -1.0. Note that positive values encourage selfing.

GenericIndividualCriterionWeight

This is an experimental specification in development and no yet tested! Do not use!

```
GenericIndividualCriterionWeight , [Value]
GenericIndividualCriterionWeight , 0.5
```

GenericIndividualCriterionWeight specifies the weight to incorporate additional individual specific criteria (GenericIndividualCriterionFile) into optimisation. Specifically, this is added to the objective function: $\sum_i^k w_i \times \mathbf{x}^T \mathbf{g}_i$, where w_i is a user defined weight (GenericIndividualCriterionWeight), $\mathbf{x}$ are proportional Contributions, and $\mathbf{g}_i$ are values for the i-th generic criterion out of k (GenericIndividualCriterionColumns).

GenericMatingCriterionWeight

This is an experimental specification in development and no yet tested! Do not use!

```
GenericMatingCriterionWeight , [Value]
GenericMatingCriterionWeight , 0.5
```

GenericMatingCriterionWeight specifies the weight to incorporate additional mating specific criteria (GenericMatingCriterionFile) into optimisation. Specifically, this is added to the objective function: $\sum_i^k w_i \times \mathbf{s}^T \mathbf{g}_i$, where w_i is a user defined weight, $\mathbf{s}$ is a vector of zeroes and $1/n_m$'s (n_m is number of matings), and $\mathbf{g}_i$ are values for the i-th generic criterion out of k (GenericMatingCriterionColumns).

EvolAlgNumberOfSolutions

```
EvolAlgNumberOfSolutions , [Value]
EvolAlgNumberOfSolutions , 100
```

EvolAlgNumberOfSolutions specifies the number of solutions to evaluate in every iteration of the evolutionary algorithm. Default is *100* for optimisations with less than 100 parameters, $max(100, 0.5 * p)$ for optimisations with more than 100 parameters and less than 1000 parameters, and *500* for optimisations with more than 1000 parameters. These defaults try to balance the amount of calculation and convergence following Chen et al. (2015).

EvolAlgNumberOfIterations

```
EvolAlgNumberOfIterations , [Value]
EvolAlgNumberOfIterations , 10000
```

EvolAlgNumberOfIterations specifies the maximum number of iterations for the evolutionary alghorithm. The default is *100 000*.

EvolAlgNumberOfIterationsStop

```
EvolAlgNumberOfIterationsStop , [Value]
EvolAlgNumberOfIterationsStop , 1000
```

EvolAlgNumberOfIterationsStop specifies the number of iterations to stop the optimisation upon no improvement of objective. Default is *100*.

EvolAlgNumberOfIterationsPrint

```
EvolAlgNumberOfIterationsPrint , [Value]
EvolAlgNumberOfIterationsPrint , 10000
```

EvolAlgNumberOfIterationsPrint specifies the number of iterations between printing successful optimisation steps. Default is *100*.

EvolAlgStopToleranceCoancestry

```
EvolAlgStopToleranceCoancestry , [Value]
EvolAlgStopToleranceCoancestry , 0.01
```

EvolAlgStopToleranceCoancestry specifies the stopping tolerance for the objective of ModeMinCoancestry and ModeMinInbreeding, i.e., when objective does not change by the tolerance value in the last EvolAlgNumberOfIterationsStop iterations, the optimisation is stopped. Default is *0.001*.

EvolAlgStopTol

```
EvolAlgStopTol , [Value]
EvolAlgStopTol , 0.01
```

EvolAlgStopTol specifies the stopping tolerance for the objective of ModeMaxCriterion and ModeOpt, i.e., when objective does not change b y the tolerance value in the last EvolAlgNumberOfIterationsStop iterations, the optimisation is stopped. Default is *0.001*.

EvolAlgLogAllSolutions

```
EvolAlgLogAllSolutions , [Yes/No]
EvolAlgLogAllSolutions , Yes
```

EvolAlgLogAllSolutions specifies whether all evaluated solutions should be logged in the output. This is useful to observe the path of optimisation (see Figure 1). Default is *No*.

EvolAlgNumberOfIterationsBurnin

```
EvolAlgNumberOfIterationsBurnin , [Value]
EvolAlgNumberOfIterationsBurnin , 1000
```

EvolAlgNumberOfIterationsBurnin specifies the number of iterations for "burn-in", where the evolutionary algorithm parameters can be specified to have different values. Default is *100*. See DiffEvolParameterCrBurnin.

Differential evolution parameters

The following specifications are for the implementation of evolutionary algorithm based on differential evolution (Storn and Price, 1997) with modifications to avoid premature convergence (Gondro and Kinghorn, 2009). The differential evolution algorithm has two key parameters: the cross-over parameter and the F parameter.

The cross-over parameter [0, 1] should be low (<=0.1) for separable problems and large (0.9) for non-separable (epistatic) problems. Note that large values encourage exploration (large moves away from the target vector), while low values encourage exploitation (small moves away from the target vector, more closer to the base vector (Montgomery and Chen, 2010).

The F parameter [0, 1+] should be 0, 2/n, or 1.2. Note that large values mean more exploration away from the base vector. The F parameters should be large for large cross-over parameter (Storn and Price, 1997; Montgomery and Chen, 2010).

DiffEvolParameterCrBurnin

```
DiffEvolParameterCrBurnin , [Value]
DiffEvolParameterCrBurnin , 0.9
```

DiffEvolParameterCrBurnin specifies the cross-over parameter of evolutionary algorithm - for the "burn-in" phase. Default is *0.9*.

DiffEvolParameterCr1

```
DiffEvolParameterCr1 , [Value]
DiffEvolParameterCr1 , 0.9
```

DiffEvolParameterCr1 specifies the cross-over parameter of evolutionary algorithm - used every non 5th iteration. Default is *0.9*.

DiffEvolParameterCr2

```
DiffEvolParameterCr2 , [Value]
DiffEvolParameterCr2 , 0.9
```

DiffEvolParameterCr2 specifies the cross-over parameter of evolutionary algorithm - used every 5th iteration. Default is *0.9*.

DiffEvolParameterFBase

```
DiffEvolParameterFBase , [Value]
DiffEvolParameterFBase , 0.1
```

DiffEvolParameterFBase specifies the F parameter of evolutionary algorithm. Default is *0.2*.

DiffEvolParameterFHigh1

```
DiffEvolParameterFHigh1 , [Value]
DiffEvolParameterFHigh1 , 1.0
```

DiffEvolParameterFHigh1 specifies the F parameter of evolutionary algorithm - used every 4th iteration. Default is *0.2*.

DiffEvolParameterFHigh2

```
DiffEvolParameterFHigh2 , [Value]
DiffEvolParameterFHigh2 , 4.0
```

DiffEvolParameterFHigh2 specifies the F parameter of evolutionary algorithm - used every 7th iteration. Default is *0.9*.

Other

OutputBasename

```
OutputBasename , [String]
OutputBasename , AnalysisX_
```

OutputBasename specifies basename prefix of all file outputs.

SeedFile

```
SeedFile , [String]
SeedFile , Seed.txt
```

SeedFile specifies a file that contains a single value used as a seed for random number generator.

Seed

```
Seed , [Value]
Seed , 0.5
```

Seed specifies a single value used as a seed for random number generator.

Stop

```
Stop
```

Stop ends the specification file. Any specifications following Stop are ignored.

Output files

The **AlphaMate** output consists of:

These outputs are stored in files, but are also printed to standard output during the program run so that users can monitor the progress of optimisation.

Summary of input data

These file report summaries of the input data: *CoancestrySummary.txt*, *InbreedingSummary.txt*, *SelCriterionSummary.txt*, *PAGESummary.txt*, *GenericIndCritSummary.txt*, *GenericMatCritSummary.txt*. Some of these files might be available only when specific input data is provided.

List of contributors with associated data

There is one contributors file for each optimisation. The files have basename *ContributorsPlanMode*, e.g., *ContributorsPlanModeMinCoancestry.txt*, *ContributorsModeOptTarget1.txt*, etc. The file reports which individuals should have non-zero contributions to the next generation, their associated data (gender, selection criterion, and average coancestry with all of the candidates), and finally how many Contributions should they have (in proportional and absolute form).

```
     Id Gender SelCriterion AvgCoancestry Contribution nMating
id1       1        17.13        0.7802         0.05      50
id2       1        16.75        0.7790         0.02      20
id3       1        16.75        0.7790         0.02      20
...
```

Mating plan

There is one mating plan file for each optimisation. The files have basename *MatingPlanMode*, e.g., *MatingPlanModeMinCoancestry.txt*, *MatingPlanModeOptTarget1.txt*, etc. The mating plan looks like this:

```
Mating Parent1 Parent2
     1 id1     id5
     2 id1     id7
     3 id2     id20
...
```

Optimisation log

There are several optimisation log files.

The files *Targets.txt* and *Frontier.txt* report different criteria as achieved by the end of each optimisation. See Definitions for the meaning of the criteria. There is one row in the files for each optimisation target.

The files with basename *OptimisationLogMode* report the optimisation path. There is one file for each optimisation target/mode, e.g., *OptimisationLogModeMinCoancestry.txt*, *OptimisationLogModeOptTarget1.txt*, etc.

The files with basename *OptimisationLogPopMode* report all the evaluated solutions during optimisation. There is one file for each optimisation target/mode, e.g., *OptimisationLogPopModeMinCoancestry.txt*, *OptimisationLogPopModeOptTarget1.txt*, etc.

Used seed

The used seed to initiate the random number generator stream is saved in the file *SeedUsed.txt*. This file can be used as input for specification SeedFile to repeat the same optimisation. Alternatively the saved seed value can be used as input for specification Seed.

Utilities

A utility R script *PlotFrontier.R* is provided to plot the *Optimisation log* outputs. Specifically it plots the Pareto frontier, the optimisation paths, and the "swarm" of all evaluated solutions during the optimisation. Figure 1 is an example output. More information about the script can be obtained by:

```
./PlotFrontier.R --help
```

Further R utilities will be provided in the next versions.

Examples

There are several examples shipped with **AlphaMate** that demonstrate its use. Each example is provided in specific folder with specifications file and data input files. All the examples are simulated so that we can distribute the data, but they do represent real-like use cases. Below we quickly summarize the examples and results.

Animal

This example demonstrates an animal population with 1000 candidates - half males and half females. The data description shows:

- mean selection criterion of 16.28
- group coancestry of 0.784
- inbreeding of 0.782

The aim is to select 25 males and mate them with the females. All selected males should be used equally. The breeder set the target to 30° and wanted to achieve 75% of minimum possible inbreeding. The results show that:

- the maximum possible selection criterion is 16.98 (standardized 1.04)
- the minimum possible group coancestry is 0.777 (rate of coancestry -0.033)
- the minimum possible inbreeding is 0.772 (rate of inbreeding -0.045)
- the balanced solution gives
  - selection criterion of 16.67 (standardized 0.58, percentage max 68)

  - group coancestry of 0.782 (rate of coancestry of -0.008, percentage min 53)
  - inbreeding of 0.776 (rate of inbreeding of -0.027, percentage min 75)

PlantCross

This example demonstrates two populations (pools) of cross-pollinating plants. The data description of both pools jointly shows:

- mean selection criterion of 17.40
- group coancestry of 0.865

The aim is to create 32 crosses between the pools. Each parent can contribute up to four crosses. The breeder set the target to 30°. The results show that:

- the maximum possible selection criterion is 18.20 (standardized 2.12)
- the minimum possible group coancestry is 0.851 (rate of coancestry -0.104)
- the balanced solution gives
  - selection criterion of 18.07 (standardized 1.76, percentage max 88)
  - group coancestry of 0.867 (rate of coancestry of 0.011, percentage min 51)

PlantSelf

This example demonstrates one plant population of self-pollinating outbred plants. The data description shows:

- mean selection criterion of 13.31
- group coancestry of 0.812

The aim is to create 32 crosses between the pools. Each parent can contribute up to ten crosses and selfing is allowed. The breeder set the target to 30°. The results show that:

- the maximum possible selection criterion is 14.19 (standardized 2.45)
- the minimum possible group coancestry is 0.801 (rate of coancestry -0.056)
- the balanced solution gives
  - selection criterion of 14.07 (standardized 2.11, percentage max 90)
  - group coancestry of 0.815 (rate of coancestry of 0.014, percentage min 52)

GenomeEditing

This example is the same as the Animal example, but attempts to optimize also which five males should be genome edited, provided that we can predict in advance their selection criterion after genome editing. The data description shows:

- mean selection criterion of 16.28
- group coancestry of 0.784

The example explores the whole Pareto frontier of genetic gain and group coancestry. Looking only at the 30° and comparing optimisation with the baseline selection criteria and selection criteria with 1, 5 or 20 genome edits we get:

- baseline
  - selection criterion of 16.76 (standardized 0.72)
  - group coancestry of 0.783 (rate of coancestry of -0.004)

- 1 genome edit
  - selection criterion of 16.77 (standardized 0.73)
  - group coancestry of 0.783 (rate of coancestry of -0.004)
- 5 genome edits
  - selection criterion of 16.85 (standardized 0.85)
  - group coancestry of 0.783 (rate of coancestry of -0.005)
- 20 genome edits
  - selection criterion of 16.99 (standardized 1.05)
  - group coancestry of 0.783 (rate of coancestry of -0.006)

References and Definitions

Definitions

Selection criterion

Selection criterion (y) is any numerical representation that a breeder can rank and select individuals on, e.g., phenotypes, estimated genetic / breeding values, etc.

See also Genetic gain and Maximum percentage gain.

Genetic gain

Genetic gain is the expected change in selection criterion between the current and the next generation. **AlphaMate** calculates it as $\mathbf{x}^T\mathbf{y}$, where $\mathbf{x}$ is a vector of proportional Contributions and $\mathbf{y}$ is a vector of selection criteria. Note that this calculation represents expected genetic gain only when the selection criteria are breeding values.

See also Selection criterion and Maximum percentage gain.

Maximum percentage gain

Maximum percentage gain is a normalized measure of genetic gain. It ranges from 0 to 100. The value of 0 is chosen to be the mean criterion achieved at minimum possible group coancestry. The value of 100 is chosen to be the mean criterion achieved with truncation selection (ignoring coancestry between individuals or inbreeding of matings). See Figure 1 for a visual representation.

See also Selection criterion and Genetic gain.

Loss of genetic diversity

Loss of genetic diversity can be measured in many ways. **AlphaMate** uses Group coancestry and increase in this metric as a proxy for the loss of genetic diversity.

Coancestry

Coancestry measures genetic identity between individuals. Specifically, it is the probability that genomes of two individuals are identical. With two genomes per individual, we have four combinations. For each combination we can evaluate the identity between the genomes, i.e., c_{i_1,j_1} is identity between the genome 1 of individual i and genome 1 of individual j. The coefficient of coancestry is then expected identity over the four combinations: $c_{i,j} = \frac{1}{4}(c_{i_1,j_1} + c_{i_1,j_2} + c_{i_2,j_1} + c_{i_2,j_2})$.

When we apply the above formulation to one individual i, we obtain $f_i = c_{i,i} = \frac{1}{4}(c_{i_1,i_1} + c_{i_1,i_2} + c_{i_2,i_1} + c_{i_2,i_2}) = \frac{1}{4}(1 + c_{i_1,i_2} + c_{i_2,i_1} + 1) = \frac{1}{2} + \frac{1}{2}f_i = \frac{1}{2}(1 + f_i)$, where f_i is the coefficient of Inbreeding.

See also Numerator relationship, Group coancestry, Rate of coancestry, Minimum percentage coancestry, and Inbreeding.

Numerator relationship

Numerator relationship coefficient is simply twice the coefficient of coancestry. It measures additive genetic covariance between individuals.

See also Coancestry, Group coancestry, Rate of coancestry, and Minimum percentage coancestry.

Group coancestry

Group coancestry measures identity between genomes of all contributors (males vs females, males vs males, and females vs females). We can calculate this measure as $c_g = \mathbf{x}^T\mathbf{C}\mathbf{x}$, where mathbf{x} is a vector of proportional Contributions and mathbf{C} is a matrix of Coancestry coefficients. If we set all Contributions to $\mathbf{x} = 1/(2m)$, then group coancestry measures coancestry between all candidates, i.e., the current group coancestry. When we use the optimised Contributions, the group coancestry measures coancestry between contributors (weighted by their proportional contributions). This in turn measures group coancestry of the next generation, because genomes of the next generation will be sampled from genomes of contributors. Note that group coancestry assumes that mating of the contributors is random, including mating among males and among females, and including selfing.

See also Coancestry, Numerator relationship, Rate of coancestry, and Minimum percentage coancestry.

Inbreeding

Inbreeding is a result of mating related individuals. A measure of inbreeding is inbreeding coefficient, which measures identity between genomes of an individual. If parents of an individual were related, then the individual has a non-zero probability that its genomes are (partly) identical. Following the same notation as for the coefficient of Coancestry, the coefficient of inbreeding is $f_i = c_{i_1,i_2}$.

Note that the coefficient of inbreeding of an individual is equal to the coefficient of Coancestry between its parents. To see this note that individuals' genomes are sampled from its parents, hence the probability that individuals' genomes are identical is the same as expected identity between genomes of its parents (=the coefficient of coancestry). **AlphaMate** uses this relationship to calculate the expected inbreeding of a mating form the coancestries between contributors.

See also Rate of inbreeding, Inbreeding depression, and Coancestry .

Inbreeding depression

Inbreeding depression is undesired reduction/increase in some traits of individuals, due to genetic "effects" that are common with inbreeding. One such "effect" is that inbreeding increases homozygosity and if an individual is homozygous for alleles that have undesired effect, then this leads to depression. The latter is particularly the case when such alleles are recessive.

See also Inbreeding.

Rate of coancestry

Rate of coancestry measures proportional increase in coancestry between generations: $\triangle C = (\bar{c}_{t+1} - \bar{c}_t)/(1 - \bar{c}_t)$. While the scale of Coancestry values depend on the type of input data and estimators, the rate of coancestry is scale free. Rate is also useful to set targets as we can use average/group coancestry in the current population $\bar{c}_t$ and the rate of coancestry $\triangle C$ to evaluate targeted coancestry in the next generation as: $\bar{c}_{t+1} = \triangle C + (1 - \triangle C) * \bar{c}_t = \triangle C * (1 - \bar{c}_t) + \bar{c}_t$.

See also Coancestry, Numerator relationship, Group coancestry, and Minimum percentage coancestry.

Rate of inbreeding

Rate of inbreeding measures proportional increase in inbreeding between generations: $\triangle F = (\overline{f}_{t+1} - \overline{f}_t)/(1-\overline{f}_t)$. While the scale of Inbreeding values depend on the type of input data and estimators, the rate of inbreeding is scale free. Rate is also useful to set targets as we can use average inbreeding in the current population $\overline{f}_t$ and the rate of inbreeding $\triangle F$ to evaluate targeted inbreeding in the next generation as: $\overline{f}_{t+1} = \triangle F + (1 - \triangle F) * \overline{f}_t = \triangle F * (1 - \overline{f}_t) + \overline{f}_t$.

See also Inbreeding and Rate of coancestry.

Minimum percentage coancestry

Minimum percentage coancestry is a normalized measure of a range of achievable Group coancestry. It ranges from 0 to 100. The value of 0 is chosen to be the group coancestry achieved with truncation selection (ignoring coancestry between individuals or inbreeding of matings). The value of 100 is chosen to be the group coancestry achieved at minimum possible group coancestry. See Figure 1 for a visual representation.

See also Coancestry, Numerator relationship, Group coancestry, and Rate of coancestry.

Minimum percentage inbreeding

Minimum percentage inbreeding is a normalized measure of a range of achievable Inbreeding. It ranges from 0 to 100. The value of 0 is chosen to be the mean inbreeding of matings with truncation selection (ignoring coancestry between individuals or inbreeding of matings). The value of 100 is chosen to be the mean inbreeding of matings achieved at minimum possible group coancestry.

See also Inbreeding and Rate of inbreeding.

Trigonometric degrees

Trigonometric degrees describe the balance between the genetic gain and loss of genetic diversity in the context of Figure 1. Specifically, they measure the angle between the truncation selection solution (the y axis) and a balanced solution, all the way to the minimum group coancestry solution (the x axis) (Kinghorn, 2011: Appendix). This formulation assumes a unit circular shape of the Pareto frontier between genetic gain and group coancestry, which seems to hold approximately (Figure 1). Trigonometric degrees are therefore a handy measure of balance between the two most important and conflicting targets in breeding programs. They also allow for simple conversion from say targeted degrees to targeted genetic gain and group coancestry and vice versa.

- A solution with 0° is the truncation selection solution that gives Maximum percentage gain of 100 and Minimum percentage coancestry of 0.
- A solution with 45° is the solution that gives Maximum percentage gain of 70.7 and Minimum percentage coancestry of 70.7.
- A solution with 90° is the minimum group coancestry solution that gives Maximum percentage gain of 0 and Minimum percentage coancestry of 100.

We can calculate trigonometric degrees from:

- Maximum percentage gain as $acos(MaxCriterionPct/100) \times 180/\pi$ or
- Minimum percentage coancestry as $asin(MinCoancestryPct/100) \times 180/\pi$.

We can calculate Maximum percentage gain from trigonometric degrees as $cos(Degree \times \pi/180) \times 100$.

We can calculate Minimum percentage coancestry from trigonometric degrees as $sin(Degree \times \pi/180) \times 100$.

With the above four formulas we can convert degrees to genetic gain and group coancestry and vice versa. **AlphaMate** uses this as a general way to convert between the various forms of specifying targeted genetic gain, targeted group coancestry, and trigonometric degrees.

Note that the trigonometric degrees only describe balance of a particular solution or set targeted balance. Evolutionary algorithm works along these targets, but can go beyond the assumed circular shape of frontier.

Contributions

A contribution measures how many times a particular individual (contributor) will mate. In **AlphaMate** we optimize absolute contributions $n = 0, 1, 2, ...$ Contributions are sometimes also presented as proportional values: $x = n/(2m)$, where m is the total number of matings.

References

- Chen, S., Montgomery, J., Bolufe-Roehler, A. (2015) Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. Appl. Intell., 42, 514. http://doi.org/10.1007/s10489-014-0613-2
- Deb, K. (2014) Multi-objective optimization. In: Search Methodologies, Springer, 403-449, http://doi.org/10.1007/978-1-4614-6940-7_15.
- Gondro, C., Kinghorn, B.P. (2009) Application of evolutionary algorithms to solve complex problems in quantitative genetics and bioinformatics. University of Guelph, 96p, http://bkinghor.une.edu.au/Evolutionary_Algorithms_CGIL2008.pdf.
- Kinghorn, B.P. (2011) An algorithm for efficient constrained mate selection. Genet. Sel. Evol., 43, http://doi.org/10.1186/1297-9686-43-4.
- Kinghorn, B.P., Shepherd, R.K. (1999) Mate selection for the tactical implementation of breeding programs. Assoc. Advmt. Anim. Breed. Genet. 13, 130-133, http://www.aaabg.org/livestocklibrary/1999/AB99025.pdf.
- Lampinen, J., Zelinka, I. (1999) Mixed variable non-linear optimization by differential evolution. Proceedings of Nostradamus, 99, 7-8, http://pdfs.semanticscholar.org/26e0/8a8d1371c51d7b1e4634e5b3f6a501e29d73.pdf.
- Montgomery, J. Chen, S. (2010) An analysis of the operation of differential evolution at high and low crossover rates. In: IEEE Congress on Evolutionary Computation, http://doi.org/10.1109/CEC.2010.5586128.
- Storn, R. Price, K., (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim., 11, 341-359, http://doi.org/10.1023/A:1008202821328.