

# Package ‘evola’

January 18, 2025

**Version** 1.0.4

**Date** 2025-01-01

**Title** Evolutionary Algorithm

**Maintainer** Giovanni Covarrubias-Pazaran <cova\_ruber@live.com.mx>

**Description** Runs a genetic algorithm using the 'AlphaSimR' machinery <[doi:10.1093/g3journal/jkaa017](https://doi.org/10.1093/g3journal/jkaa017)> and the coalescent simulator 'MaCS' <[doi:10.1101/gr.083634.108](https://doi.org/10.1101/gr.083634.108)>.

**Depends** R(>= 3.5.0), AlphaSimR (>= 1.4.2), Matrix (>= 1.0), methods, crayon

**LazyLoad** yes

**LazyData** yes

**License** GPL (>= 2)

**NeedsCompilation** yes

**Author** Giovanni Covarrubias-Pazaran [aut, cre]  
(<<https://orcid.org/0000-0002-7194-3837>>)

**Repository** CRAN

**Suggests** rmarkdown, knitr

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Date/Publication** 2025-01-18 22:20:01 UTC

## Contents

evola-package . . . . .	2
A.mat . . . . .	3
bestSol . . . . .	4
DT_cpdata . . . . .	6
DT_technow . . . . .	7
DT_wheat . . . . .	9
evolafit . . . . .	11
fitnessf . . . . .	16

Jc . . . . .	17
Jr . . . . .	18
overlay . . . . .	19
pareto . . . . .	20
pmonitor . . . . .	21
stan . . . . .	21
varM . . . . .	22

<b>Index</b>	<b>24</b>
--------------	-----------

---

evola-package	<b>EVOLutionary Algorithm</b>
---------------	-------------------------------

---

## Description

The evola package is nice wrapper of the AlphaSimR package that enables the use of the evolutionary algorithm to solve complex questions in a simple form.

The `evolafit` function is the core function of the package which allows the user to specify the problem and constraints to find a close-to-optimal solution using the evolutionary forces.

## Keeping evola updated

The evola package is updated on CRAN every 4-months due to CRAN policies but you can find the latest source at <https://github.com/covaruber/evola>. This can be easily installed typing the following in the R console:

```
library(devtools)
install_github("covaruber/evola")
```

This is recommended if you reported a bug, was fixed and was immediately pushed to GitHub but not in CRAN until the next update.

## Tutorials

For tutorials on how to perform different analysis with evola please look at the vignettes by typing in the terminal:

```
vignette("evola.intro")
```

## Getting started

The package has been equipped with a couple of datasets to learn how to use the evola package:

- \* `DT_technow` dataset to perform optimal cross selection.
- \* `DT_wheat` dataset to perform optimal training population selection.
- \* `DT_cpdata` dataset to perform optimal individual.

## Models Enabled

The machinery behind the scenes is AlphaSimR.

**Bug report and contact**

If you have any questions or suggestions please post it in <https://stackoverflow.com> or <https://stats.stackexchange.com>

I'll be glad to help or answer any question. I have spent a valuable amount of time developing this package. Please cite this package in your publication. Type 'citation("evola")' to know how to cite it.

**Author(s)**

Giovanny Covarrubias-Pazaran

**References**

Giovanny Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to *Bioinformatics*.

Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. *G3 Gene|Genomes|Genetics* 11(2):jkaa017. <https://doi.org/10.1093/g3journal/jkaa017>.

Chen GK, Marjoram P, Wall JD (2009). Fast and Flexible Simulation of DNA Sequence Data. *Genome Research*, 19, 136-142. <http://genome.cshlp.org/content/19/1/136>.

---

A.mat

*Additive relationship matrix*


---

**Description**

Calculates the realized additive relationship matrix.

**Usage**

```
A.mat(X,min.MAF=NULL)
```

**Arguments**

X	Matrix ( $n \times m$ ) of unphased genotypes for $n$ lines and $m$ biallelic markers, coded as $\{-1,0,1\}$ . Fractional (imputed) and missing values (NA) are allowed.
min.MAF	Minimum minor allele frequency. The A matrix is not sensitive to rare alleles, so by default only monomorphic markers are removed.

**Details**

For vanraden method: the marker matrix is centered by subtracting column means  $M = X - ms$  where  $ms$  is the column means. Then  $A = MM'/c$ , where  $c = \sum_k d_k/k$ , the mean value of the diagonal values of the  $MM'$  portion.

**Value**

If `return.imputed = FALSE`, the  $n \times n$  additive relationship matrix is returned.

If `return.imputed = TRUE`, the function returns a list containing

**\$A** the A matrix

**References**

Giovanni Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to Bioinformatics.

**See Also**

[evolafit](#) – the core function of the package

**Examples**

```
## random population of 200 lines with 1000 markers
X <- matrix(rep(0,200*1000),200,1000)
for (i in 1:200) {
  X[i,] <- ifelse(runif(1000)<0.5,-1,1)
}

A <- A.mat(X)

## take a look at the Genomic relationship matrix
colfunc <- colorRampPalette(c("steelblue4","springgreen","yellow"))
hv <- heatmap(A[1:15,1:15], col = colfunc(100), Colv = "Rowv")
str(hv)
```

---

bestSol

*Extract the index of the best solution*

---

**Description**

Extracts the index of the best solution for all traits under the constraints specified.

**Usage**

```
bestSol(object, selectTop=TRUE)
```

**Arguments**

<code>object</code>	A resulting object from the function <code>evolafit</code> .
<code>selectTop</code>	Selects highest values for the fitness value if <code>TRUE</code> . Selects lowest values if <code>FALSE</code> .

## Details

A simple apply function looking at the fitness value of all the solution in the last generation to find the maximum value.

## Value

**\$res** the vector of best solutions in M for each trait in the problem

## References

Giovanny Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to *Bioinformatics*.

## See Also

[evolafit](#) – the core function of the package

## Examples

```
set.seed(1)
# Data
Gems <- data.frame(
  Color = c("Red", "Blue", "Purple", "Orange",
            "Green", "Pink", "White", "Black",
            "Yellow"),
  Weight = round(runif(9,0.5,5),2),
  Value = round(abs(rnorm(9,0,5))+0.5,2),
  Times=c(rep(1,8),0)
)
head(Gems)

# Task: Gem selection.
# Aim: Get highest combined value.
# Restriction: Max weight of the gem combined = 10.
res0<-evolafit(cbind(Weight,Value)~Color, dt= Gems,
  # constraints: if greater than this ignore
  constraintsUB = c(10,Inf),
  # constraints: if smaller than this ignore
  constraintsLB= c(-Inf,-Inf),
  # weight the traits for the selection
  traitWeight = c(0,1),
  # population parameters
  nCrosses = 100, nProgeny = 20, recombGens = 1,
  # coancestry parameters
  A=NULL, lambda=c(0,0), nQTLperInd = 1,
  # selection parameters
  propSelBetween = .9, propSelWithin =0.9,
  nGenerations = 50
)
```

```
bestSol(res0)
```

---

DT\_cpdata

*Genotypic and Phenotypic data for a CP population*


---

## Description

A CP population or F1 cross is the designation for a cross between 2 highly heterozygote individuals; i.e. humans, fruit crops, breeding populations in recurrent selection.

This dataset contains phenotypic data for 363 siblings for an F1 cross. These are averages over 2 environments evaluated for 4 traits; color, yield, fruit average weight, and firmness. The columns in the CPgeno file are the markers whereas the rows are the individuals. The CPpheno data frame contains the measurements for the 363 siblings, and as mentioned before are averages over 2 environments.

## Usage

```
data("DT_cpdata")
```

## Format

The format is: chr "DT\_cpdata"

## Source

This data was simulated for fruit breeding applications.

## References

Giovanny Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to Bioinformatics.

Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. *G3 Gene|Genomes|Genetics* 11(2):jkaa017. <https://doi.org/10.1093/g3journal/jkaa017>.

Chen GK, Marjoram P, Wall JD (2009). Fast and Flexible Simulation of DNA Sequence Data. *Genome Research*, 19, 136-142. <http://genome.cshlp.org/content/19/1/136>.

## Examples

```
data(DT_cpdata)
DT <- DT_cpdata

# get best 20 individuals weighting variance by ~0.5=(30*pi)/180
res<-evolafit(formula=cbind(Yield, occ)~id, dt= DT,
              # constraints: if sum is greater than this ignore
              constraintsUB = c(Inf,20),
```

```

# constraints: if sum is smaller than this ignore
constraintsLB= c(-Inf,-Inf),
# weight the traits for the selection
traitWeight = c(1,0),
# population parameters
nCrosses = 100, nProgeny = 10,
recombGens=1, nChr=1, mutRate=0,
# coancestry parameters
A=A, lambda= (30*pi)/180 , nQTLperInd = 2,
# selection parameters
propSelBetween = 0.5, propSelWithin =0.5,
nGenerations = 40, keepBest=FALSE)

best = bestSol(res)["pop", "Yield"];best
xa = (res$M %%% DT$Yield)[best,]; xa
xAx = res$M[best,] %%% A %%% res$M[best,]; xAx
sum(res$M[best,]) # total # of inds selected

pmonitor(res)

plot(DT$Yield, col=as.factor(res$M[best,]),
     pch=(res$M[best,]*19)+1)

pareto(res)

```

---

DT\_technow

*Genotypic and Phenotypic data from single cross hybrids (Technow et al., 2014)*


---

## Description

This dataset contains phenotypic data for 2 traits measured in 1254 single cross hybrids coming from the cross of Flint x Dent heterotic groups. In addition contains the genotypic data (35,478 markers) for each of the 123 Dent lines and 86 Flint lines. The purpose of this data is to demonstrate the prediction of unrealized crosses (9324 unrealized crosses, 1254 evaluated, total 10578 single crosses). We have added the additive relationship matrix (A) but can be easily obtained using the A.mat function on the marker data. Please if using this data for your own research cite Technow et al. (2014) publication (see References).

## Usage

```
data("DT_technow")
```

## Format

The format is: chr "DT\_technow"

## Source

This data was extracted from Technow et al. (2014).

## References

If using this data for your own research please cite:

Technow et al. 2014. Genome properties and prospects of genomic predictions of hybrid performance in a Breeding program of maize. *Genetics* 197:1343-1355.

Giovanny Covarrubias-Pazarán (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to *Bioinformatics*.

Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. *G3 Genes|Genomes|Genetics* 11(2):jkaa017. <https://doi.org/10.1093/g3journal/jkaa017>.

Chen GK, Marjoram P, Wall JD (2009). Fast and Flexible Simulation of DNA Sequence Data. *Genome Research*, 19, 136-142. <http://genome.cshlp.org/content/19/1/136>.

## Examples

```
data(DT_technow)
DT <- DT_technow
DT$occ <- 1; DT$occ[1]=0
M <- M_technow

A <- A.mat(M)
# run the genetic algorithm
# we assign a weight to x'Ax of (20*pi)/180=0.34
res<-evolafit(formula = c(GY, occ)~hy,
              dt= DT,
              # constraints: if sum is greater than this ignore
              constraintsUB = c(Inf,100),
              # constraints: if sum is smaller than this ignore
              constraintsLB= c(-Inf,-Inf),
              # weight the traits for the selection
              traitWeight = c(1,0),
              # population parameters
              nCrosses = 100, nProgeny = 10,
              recombGens=1, nChr=1, mutRate=0,
              # coancestry parameters
              A=A, lambda= (20*pi)/180 , nQTLperInd = 70,
              # selection parameters
              propSelBetween = 0.5, propSelWithin =0.5,
              nGenerations = 20, keepBest=FALSE)

best = bestSol(res)["pop", "GY"]
xa = (res$M %*% DT$GY)[best,]; xa
xAX = res$M[best,] %*% A %*% res$M[best,]; xAX
sum(res$M[best,]) # total # of inds selected

pmonitor(res)
plot(DT$GY, col=as.factor(res$M[best,]),
```



```
pch=(res$M[best,]*19)+1)
pareto(res)
```

---

DT\_wheat

*wheat lines dataset*

---

## Description

Information from a collection of 599 historical CIMMYT wheat lines. The wheat data set is from CIMMYT's Global Wheat Program. Historically, this program has conducted numerous international trials across a wide variety of wheat-producing environments. The environments represented in these trials were grouped into four basic target sets of environments comprising four main agroclimatic regions previously defined and widely used by CIMMYT's Global Wheat Breeding Program. The phenotypic trait considered here was the average grain yield (GY) of the 599 wheat lines evaluated in each of these four mega-environments.

A pedigree tracing back many generations was available, and the Browse application of the International Crop Information System (ICIS), as described in (McLaren *et al.* 2000, 2005) was used for deriving the relationship matrix A among the 599 lines; it accounts for selection and inbreeding.

Wheat lines were recently genotyped using 1447 Diversity Array Technology (DArT) generated by Triticaret Pty. Ltd. (Canberra, Australia; <http://www.triticarte.com.au>). The DArT markers may take on two values, denoted by their presence or absence. Markers with a minor allele frequency lower than 0.05 were removed, and missing genotypes were imputed with samples from the marginal distribution of marker genotypes, that is,  $x_{ij} = \text{Bernoulli}(\hat{p}_j)$ , where  $\hat{p}_j$  is the estimated allele frequency computed from the non-missing genotypes. The number of DArT MMs after edition was 1279.

## Usage

```
data(DT_wheat)
```

## Format

Matrix Y contains the average grain yield, column 1: Grain yield for environment 1 and so on.

## Source

International Maize and Wheat Improvement Center (CIMMYT), Mexico.

## References

- Giovanny Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to *Bioinformatics*.
- Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. *G3 GeneGenomesGenetics* 11(2):jkaa017. <https://doi.org/10.1093/g3journal/jkaa017>.
- Chen GK, Marjoram P, Wall JD (2009). Fast and Flexible Simulation of DNA Sequence Data. *Genome Research*, 19, 136-142. <http://genome.cshlp.org/content/19/1/136>.
- McLaren, C. G., L. Ramos, C. Lopez, and W. Eusebio. 2000. "Applications of the genealogy management system." In *International Crop Information System. Technical Development Manual, version VI*, edited by McLaren, C. G., J.W. White and P.N. Fox. pp. 5.8-5.13. CIMMYT, Mexico: CIMMYT and IRRI.
- McLaren, C. G., R. Bruskiewich, A.M. Portugal, and A.B. Cosico. 2005. The International Rice Information System. A platform for meta-analysis of rice crop data. *Plant Physiology* **139**: 637-642.

## Examples

```
# example to optimize a training pop for a validation pop
data(DT_wheat)
DT <- as.data.frame(DT_wheat)
DT$id <- rownames(DT) # IDs
DT$occ <- 1; DT$occ[1]=0 # to track occurrences
DT$dummy <- 1; DT$dummy[1]=0 # dummy trait

# if genomic
GT <- GT_wheat + 1; rownames(GT) <- rownames(DT)
G <- GT%*%t(GT)
G <- G/mean(diag(G))
# if pedigree
A <- A_wheat
A[1:4,1:4]
##Perform eigenvalue decomposition for clustering
##And select cluster 5 as target set to predict
pcNum=25
svdWheat <- svd(A, nu = pcNum, nv = pcNum)
PCWheat <- A %*% svdWheat$v
rownames(PCWheat) <- rownames(A)
DistWheat <- dist(PCWheat)
TreeWheat <- cutree(hclust(DistWheat), k = 5 )
plot(PCWheat[,1], PCWheat[,2], col = TreeWheat,
     pch = as.character(TreeWheat), xlab = "pc1", ylab = "pc2")
vp <- rownames(PCWheat)[TreeWheat == 3]; length(vp)
tp <- setdiff(rownames(PCWheat),vp)

As <- A[tp,tp]
DT2 <- DT[rownames(As),]
DT2$cov <- apply(A[tp,vp],1,mean)

head(DT2)
```

```

# we assign a weight to x'Ax of (60*pi)/180=1
res<-evolafit(formula=cbind(cov, occ)~id, dt= DT2,
               # constraints: if sum is greater than this ignore
               constraintsUB = c(Inf, 100),
               # constraints: if sum is smaller than this ignore
               constraintsLB= c(-Inf, -Inf),
               # weight the traits for the selection
               traitWeight = c(1,0),
               # population parameters
               nCrosses = 100, nProgeny = 10,
               recombGens=1, nChr=1, mutRate=0,
               # coancestry parameters
               A=As, lambda= (60*pi)/180 , nQTLperInd = 80,
               # selection parameters
               propSelBetween = 0.5, propSelWithin =0.5,
               nGenerations = 30, verbose = TRUE, keepBest=FALSE)

best <- bestSol(res)["pop","cov"]
sum(res$M[best,]) # total # of inds selected
pareto(res)

```

---

evolafit

*Fits a genetic algorithm for a set of traits and constraints.*


---

## Description

Using the AlphaSimR machinery it recreates the evolutionary forces applied to a problem where possible solutions replace individuals and combinations of variables to optimize in the problem replace the genes or QTLs. Then evolutionary forces (mutation, selection and drift) are applied to find a close-to-optimal solution.

## Usage

```

evolafit(formula, dt,
          constraintsUB, constraintsLB, traitWeight,
          nCrosses=50, nProgeny=20, nGenerations=20,
          recombGens=1, nChr=1, mutRate=0,
          nQTLperInd=NULL, A=NULL, lambda=NULL,
          propSelBetween=1, propSelWithin=0.5,
          fitnessf=NULL, verbose=TRUE, dateWarning=TRUE,
          selectTop=TRUE, tolVarG=1e-6, keepBest=FALSE,
          ...)

```

**Arguments**

formula	Formula of the form $y \sim x$ where 'y' refers to the traits or features defining the average allelic substitution effects of the QTLs, and 'x' refers to the variable defining the genes or QTLs to be combined in the possible solutions.
dt	A dataset containing the features/traits and classifiers/genes/QTLs.
constraintsUB	A numeric vector specifying the upper bound constraints in the traits/features (y). The length is equal to the number of traits/features in the formula. If missing is assume an infinite value for all traits. Solutions with higher value than the upper bound are assigned a -infinite value if the argument <code>selectTop=TRUE</code> and to +infinite when <code>selectTop=FALSE</code> , which is equivalent to reject/discard a solution based on the fitness function.
constraintsLB	A numeric vector specifying the lower bound constraints in the traits/features (y). The length is equal to the number of traits/features in the formula. If missing is assume a -infinite value for all traits. Solutions with lower value than the lower bound are assigned a +infinite value if the argument <code>selectTop=TRUE</code> and to -infinite when <code>selectTop=FALSE</code> , which is equivalent to reject/discard a solution based on the fitness function.
traitWeight	A numeric vector specifying the weights that each trait has in the fitness function (e.g., a selection index). The length is equal to the number of traits/features. If missing is assumed equal weight (1) for all traits.
nCrosses	A numeric value indicating how many crosses should occur in the population of solutions at every generation.
nProgeny	A numeric value indicating how many progeny (solutions) each cross should generate in the population of solutions at every generation.
nGenerations	The number of generations that the evolutionary process should run for.
recombGens	The number of recombination generations that should occur before selection is applied. This is in case the user wants to allow for more recombination before selection operates. The default is 1.
nChr	The number of chromosomes where features/genes should be allocated to. The default value is 1 but this number can be increased to mimic more recombination events at every generation and avoid linkage disequilibrium.
mutRate	A value between 0 and 1 to indicate the proportion of random QTLs that should mutate in each individual. For example, a value of 0.1 means that a random 10% of the QTLs will mutate in each individual randomly taking values of 0 or 1. Is important to notice that this implies that the stopping criteria based in variance will never be reached because we keep introducing variance through random mutation.
nQTLperInd	The number of QTLs/genes (classifier x) that should be fixed for the positive allele at the beginning of the simulation. If not specified it will be equal to the 20% of the QTLs (calculated as the number of rows in the dt argument over 5). See details section.
A	A relationship matrix between the levels of the 'x' variable or QTLs (not between the solutions). It is a kind of a linkage disequilibrium matrix. This function can be used or ignored in the fitness function. By default the weight to

	the $x'Ax$ component is 0 though the lambda argument, where x is an individual/solution.
lambda	A numeric value indicating the weight assigned to the relationship between levels of the classifiers in comparison with the trait value. If not specified is assumed to be 0. This can be used or ignored in your customized fitness function.
propSelBetween	A numeric value between 0 and 1 indicating the proportion of families/crosses of solutions/individuals that should be selected. The default is 1, meaning all crosses are selected or passed to the next generation.
propSelWithin	A numeric value between 0 and 1 indicating the proportion of individuals/solutions within families/crosses that should be selected. The default value is 0.5, meaning that 50% of the top individuals are selected.
fitnessf	An alternative fitness function for a linear combination of the traits specified in 'y' in the formula argument. If NULL, the default function will be: $\text{function}(Y,b,d,Q)\{(Y\%*\%b) - d\}$ where $Y\%*\%b$ is equivalent to $xa$ in contribution theory, and $d$ is equal to $x'Ax$ , being $x$ the contribution vector to the solution, $a$ are the QTL effects, and $A$ is the covariance between QTLs, $Q$ is the QTL matrix for the solution, $b$ are the weights for each trait, $Y$ is the matrix of trait-values for each solution. If you provide your own fitness function please keep in mind that the variables $Y$ , $b$ , $d$ and $Q$ are already reserved and should always be added to your function (even if not used) in addition to your new variables.
verbose	A logical value indicating if we should print logs.
dateWarning	A logical value indicating if you should be warned when there is a new version on CRAN.
selectTop	Selects highest values for the fitness value if TRUE. Selects lowest values if FALSE.
tolVarG	A stopping criteria (tolerance for genetic variance) when the variance across traits is smaller than this value, which is equivalent to assume that all solutions having the same QTL profile (depleted variance). The default value is $1e-6$ and is computed as the sum of the diagonal values of the genetic variance covariance matrix between traits.
keepBest	A TRUE/FALSE value to indicate if we should store the QTL matrix and pedigree of the solutions selected across generations. This can be useful if we want to recreate the path to the best solution (e.g., best crossing path to a product).
...	Further arguments to be passed to the fitness function.

## Details

Using the AlphaSimR machinery (runMacs) it recreates the evolutionary forces applied to a problem where possible solutions replace individuals and combinations of variables in the problem replace the genes. Then evolutionary forces are applied to find a close-to-optimal solution. The number of solutions are controlled with the nCrosses and nProgeny parameters, whereas the number of initial QTLs activated in a solution is controlled by the nQTLperInd parameter. The number of activated QTLs of course will increase if has a positive effect in the fitness of the solutions. The drift force can be controlled by the recombGens parameter. The mutation rate can be controlled with the mutRate parameter. The recombination rate can be controlled with the nChr argument.

The `indivPerformance` output contains the columns `x'a`, `x'Ax`, `deltaC` and `nQTLs`. These mean the following:

In `x'a` or `score` : `'x'` represents the contribution vector, `'a'` is the vector of QTN effects, so `x'a` is the breeding value of a solution. Internally, we multiply all the matrix of solutions (Q) by the QTN effects (a). In `deltaC` : it represents the change in coancestry (e.g., inbreeding), it can be thought as the rate of coancestry. In `x'Ax` : `'x'` represents the contribution vector, `'A'` is the linkage disequilibrium matrix between QTNs (whatever the QTNs represent for your specific problem). In practice we do `QAQ'` and extract the diagonal values. In `generation` : it represents the generation at which this solution appeared. In `nQTLs` : it represent the final number of QTNs that are activated in homozygote state for the positive effect.

## Value

**indivPerformance** the matrix of `x'a` (score), `deltaC`, `x'Ax`, `generation`, `nQTLs` per solution per generation. See details section above.

**pedBest** if the argument `keepBest=TRUE` this contains the pedigree of the selected solutions across iterations.

**\$M** the matrix of QTN genotypes for the solutions/individuals selected at the end of the run.

**\$Mb** the matrix of top (parents) QTN genotypes for the solutions/individuals at the end of the run.

**\$score** a matrix with scores for different metrics across n generations of evolution.

**\$pheno** the matrix of phenotypes of individuals/solutions present in the last generation.

**\$phenoBest** the matrix of phenotypes of top (parents) individuals/solutions present in the last generation.

**pop** AlphaSimR object used for the evolutionary algorithm at the last iteration.

**best** AlphaSimR object corresponding to the best parental haplotypes/solutions selected for new crosses across all cycles.

**constCheckUB** A matrix with as many rows as solutions and columns as traits to be constrained. 0s indicate that such trait went beyond the bound in that particular solution.

**constCheckLB** A matrix with as many rows as solutions and columns as traits to be constrained. 0s indicate that such trait went beyond the bound in that particular solution.

**traits** a character vector corresponding to the name of the variables used in the fitness function.

## References

Giovanny Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to *Bioinformatics*.

Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. *G3 Gene|Genomes|Genetics* 11(2):jkaa017. <https://doi.org/10.1093/g3journal/jkaa017>.

Chen GK, Marjoram P, Wall JD (2009). Fast and Flexible Simulation of DNA Sequence Data. *Genome Research*, 19, 136-142. <http://genome.cshlp.org/content/19/1/136>.

## See Also

[evolafit](#) – the information of the package

**Examples**

```

set.seed(1)

# Data
Gems <- data.frame(
  Color = c("Red", "Blue", "Purple", "Orange",
            "Green", "Pink", "White", "Black",
            "Yellow"),
  Weight = round(runif(9,0.5,5),2),
  Value = round(abs(rnorm(9,0,5))+0.5,2),
  Times=c(rep(1,8),0)
)
head(Gems)
#   Color Weight Value
# 1  Red   4.88  9.95
# 2  Blue  1.43  2.73
# 3 Purple 1.52  2.60
# 4 Orange 3.11  0.61
# 5  Green 2.49  0.77
# 6  Pink  3.53  1.99
# 7  White 0.62  9.64
# 8  Black 2.59  1.14
# 9 Yellow 1.77 10.21

# Task: Gem selection.
# Aim: Get highest combined value.
# Restriction: Max weight of the gem combined = 10.

# simple specification
res00<-evolafit(formula=cbind(Weight,Value)~Color, dt= Gems,
                # constraints on traits: if greater than this ignore
                constraintsUB = c(10,Inf), nGenerations = 10
)
best = bestSol(res00)["pop", "Value"]
xa = res00$M[best,] %*% as.matrix(Gems[,c("Weight", "Value")]); xa
res00$M[best,]
res00$score[nrow(res00$score),]

# more complete specification
res00<-evolafit(formula=cbind(Weight,Value)~Color, dt= Gems,
                # constraints on traits: if greater than this ignore
                constraintsUB = c(10,Inf),
                # constraints on traits: if smaller than this ignore
                constraintsLB= c(-Inf,-Inf),
                # weight the traits for the selection (fitness function)
                traitWeight = c(0,1),
                # population parameters
                nCrosses = 100, nProgeny = 20,
                # genome parameters
                recombGens = 1, nChr=1, mutRate=0, nQTLperInd = 1,

```

```

# coancestry parameters
A=NULL, lambda=0,
# selection parameters
propSelBetween = .9, propSelWithin =0.9,
nGenerations = 50
)

best = bestSol(res0)["pop", "Value"]
xa = res0$M[best,] %*% as.matrix(Gems[,c("Weight", "Value")]); xa
res0$M[best,]
res0$score[nrow(res0$score),]

# $`Genes`
# Red   Blue Purple Orange  Green  Pink  White  Black Yellow
# 1     1     0     0     1     0     0     1     0
#
# $Result
# Weight Value
# 8.74  32.10
pmonitor(res0)
pareto(res0)

```

---

fitnessf

*Fitness function from contribution theory*


---

## Description

Simple function for fitness where an index of traits is weighted by the group relationship.

## Usage

```
fitnessf(Y,b,d,Q)
```

## Arguments

Y	A matrix of trait values. See details.
b	A vector of trait weights. See details.
d	A vector of group relationships. See details.
Q	A QTL matrix. See details.

## Details

A simple apply function of a regular index weighted by a vector of relationships.

$Y \% \% b - d$

Internally, we use this function in the following way:



The Y matrix is the matrix of trait-GEBVs and b is the user-specified trait weights.

$d = xAx * \lambda$ ; where  $xAx$  is equal to `Matrix::diag(Q%*%Matrix::tcrossprod(A,Q))` of dimensions  $n \times n$

Notice that Q represents the marker of QTLs (columns) for all solutions (rows) and A the LD between QTLs. The user can modify this function as needed and provide it to the `evolafit` function along with other arguments.

### Value

**\$res** a vector of values

### References

Giovanni Covarrubias-Pazarán (2024). `evola`: a simple evolutionary algorithm for complex problems. To be submitted to *Bioinformatics*.

### See Also

[evolafit](#) – the core function of the package

### Examples

```
Y=matrix(1:12,4,3) # 4 solutions with 3 traits
b=c(1,2,3) # weights for each trait
d=c(1,1,1,1) # coancestry for each solution
fitnessf(Y,b,d) # Yb - d where d is QAQ' and A is the LD between QTLs
```

---

Jc

*Matrix of ones*

---

### Description

Makes a matrix of ones with a single row and `nc` columns.

### Usage

```
Jc(nc)
```

### Arguments

`nc`                      Number of columns to create.

### Details

A simple apply function to make a matrix of one row and `nc` columns.

**Value**

**\$res** a matrix

**References**

Giovanny Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to Bioinformatics.

**See Also**

[evolafit](#) – the core function of the package

**Examples**

Jc(5)

---

Jr

*Matrix of ones*

---

**Description**

Makes a matrix of ones with a single column and nr rows.

**Usage**

Jr(nr)

**Arguments**

nr                      Number of rows to create.

**Details**

A simple apply function to make a matrix of one column and nr rows.

**Value**

**\$res** a matrix

**References**

Giovanny Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to Bioinformatics.

**See Also**

[evolafit](#) – the core function of the package

**Examples**

```
Jr(5)
```

---

overlay	<i>Overlay Matrix</i>
---------	-----------------------

---

**Description**

‘overlay’ adds *r* times the design matrix for model term *t* to the existing design matrix. Specifically, if the model up to this point has *p* effects and *t* has *a* effects, the *a* columns of the design matrix for *t* are multiplied by the scalar *r* (default value 1.0). This can be used to force a correlation of 1 between two terms as in a diallel analysis.

**Usage**

```
overlay(..., rlist=NULL, prefix=NULL, sparse=FALSE)
```

**Arguments**

<code>...</code>	as many vectors as desired to overlay.
<code>rlist</code>	a list of scalar values indicating the times that each incidence matrix overlaid should be multiplied by. By default <i>r</i> =1.
<code>prefix</code>	a character name to be added before the column names of the final overlay matrix. This may be useful if you have entries with names starting with numbers which programs such as <code>asreml</code> doesn’t like, or for posterior extraction of parameters, that way <code>'grep'</code> ing is easier.
<code>sparse</code>	a TRUE/FALSE statement specifying if the matrices should be built as sparse or regular matrices.

**Value**

**\$\$S3** an incidence matrix with as many columns levels in the vectors provided to build the incidence matrix.

**Author(s)**

Giovanny Covarrubias-Pazaran

**References**

Fikret Isik. 2009. Analysis of Diallel Mating Designs. North Carolina State University, Raleigh, USA.

Covarrubias-Pazaran G (2016) Genome assisted prediction of quantitative traits using the R package `soevolafit`. PLoS ONE 11(6): doi:10.1371/journal.pone.0156744

**See Also**

The core functions of the package [evolafit](#).

**Examples**

```
#####
#### For CRAN time limitations most lines in the
#### examples are silenced with one '#' mark,
#### remove them and run the examples
#####
data("DT_technow")
DT <- DT_technow
head(DT)
DT$dentf <- as.factor(DT$dent)
DT$flintf <- as.factor(DT$flint)

with(DT, overlay(dentf,flintf, sparse = TRUE))
with(DT, overlay(dentf,flintf, sparse = FALSE))
```

---

pareto

*plot the change of values across iterations*

---

**Description**

plot for monitoring.

**Usage**

```
pareto(object, scaled=TRUE,pch=20, xlim, ...)
```

**Arguments**

object	model object of class "evolafit"
scaled	a logical value to specify the scale of the y-axis (gain in merit).
pch	symbol for plotting points as described in par
xlim	upper and lower bound in the x-axis
...	Further arguments to be passed to the plot function.

**Value**

vector of plot

**Author(s)**

Giovanny Covarrubias

**See Also**[plot](#), [evolafit](#)

---

pmonitor	<i>plot the change of values across iterations</i>
----------	--

---

**Description**

plot for monitoring.

**Usage**

```
pmonitor(object, ...)
```

**Arguments**

object	model object of class "evolafit"
...	Further arguments to be passed to the plot function.

**Value**

vector of plot

**Author(s)**

Giovanny Covarrubias

**See Also**[plot](#), [evolafit](#)

---

stan	<i>Standardize a vector of values in range 0 to 1</i>
------	---

---

**Description**

Simple function to map a vector of values to the range of 0 and 1 values to have a better behavior of the algorithm.

**Usage**

```
stan(x)
```

**Arguments**

x	A vector of numeric values.
---	-----------------------------

**Details**

Simple function to map a vector of values to the range of 0 and 1 values to have a better behavior of the algorithm.

**Value**

**\$res** new values in range 0 to 1

**References**

Giovanny Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to *Bioinformatics*.

**See Also**

[evolafit](#) – the core function of the package

**Examples**

```
x <- rnorm(20, 10, 3);x
stan(x)
```

---

varM

*Extract the variance existing in the genome solutions*

---

**Description**

Extracts the variance found across the M element of the resulting object of the `evolafit()` function which contains the different solution and somehow represents the genome of the population.

**Usage**

```
varM(object)
```

**Arguments**

**object**            A resulting object from the function `evolafit`.

**Details**

A simple apply function looking at the variance in each column of the M element of the resulting object of the `evolafit` function.

**Value**

**\$res** a value of variance

## References

Giovanny Covarrubias-Pazaran (2024). *evola*: a simple evolutionary algorithm for complex problems. To be submitted to *Bioinformatics*.

## See Also

[evolafit](#) – the core function of the package

## Examples

```
set.seed(1)
# Data
Gems <- data.frame(
  Color = c("Red", "Blue", "Purple", "Orange",
            "Green", "Pink", "White", "Black",
            "Yellow"),
  Weight = round(runif(9,0.5,5),2),
  Value = round(abs(rnorm(9,0,5))+0.5,2),
  Times=c(rep(1,8),0)
)
head(Gems)

# Task: Gem selection.
# Aim: Get highest combined value.
# Restriction: Max weight of the gem combined = 10.
res0<-evolafit(cbind(Weight,Value)~Color, dt= Gems,
  # constraints: if greater than this ignore
  constraintsUB = c(10,Inf),
  # constraints: if smaller than this ignore
  constraintsLB= c(-Inf,-Inf),
  # weight the traits for the selection
  traitWeight = c(0,1),
  # population parameters
  nCrosses = 100, nProgeny = 20, recombGens = 1,
  # coancestry parameters
  A=NULL, lambda=c(0,0), nQTLperInd = 1,
  # selection parameters
  propSelBetween = .9, propSelWithin =0.9,
  nGenerations = 50
)

varM(res0)
```

# Index

## \* R package

evola-package, 2

## \* datasets

DT\_cpdata, 6

DT\_technow, 7

DT\_wheat, 9

## \* models

pareto, 20

pmonitor, 21

A (DT\_cpdata), 6

A.mat, 3

A\_wheat (DT\_wheat), 9

bestSol, 4

DT\_cpdata, 2, 6

DT\_technow, 2, 7

DT\_wheat, 2, 9

evola (evola-package), 2

evola-package, 2

evolafit, 2, 4, 5, 11, 14, 17, 18, 20–23

fitnessf, 16

GT\_wheat (DT\_wheat), 9

Jc, 17

Jr, 18

M\_technow (DT\_technow), 7

overlay, 19

pareto, 20

plot, 21

pmonitor, 21

stan, 21

varM, 22