Package 'evola'

August 28, 2024

Version 1.0.1

Date 2024-08-23

Title Evolutionary Algorithm

Maintainer Giovanny Covarrubias-Pazaran <cova_ruber@live.com.mx>

Description Runs a genetic algorithm using the 'AlphaSimR' machinery <doi:10.1093/g3journal/jkaa017> and the coalescent simulator 'MaCS' <doi: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 Giovanny Covarrubias-Pazaran [aut, cre] (<https://orcid.org/0000-0002-7194-3837>)

Repository CRAN

Suggests rmarkdown, knitr

VignetteBuilder knitr

Config/testthat/edition 3

Date/Publication 2024-08-28 08:40:02 UTC

Contents

evola-package 2
A.mat 3
bestSol 4
DT_cpdata 6
DT_technow 7
DT_wheat 9
evolafit 11
pareto 14

pmonitor 15
varM 16

Index **18**

`evola-package` **EVOL***utionary* **A***lgorithm*

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 manner.

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 equiped with several datasets to learn how to use the evola package:

* `DT_technow` datasets to perform optimal cross selection.

* `DT_wheat` TBD.

* `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 coumn 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

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

```
## 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)
```

Arguments

`object`	A resulting object from the function evolafit.

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, bredding populations in recurrent selection.

This dataset contains phenotpic 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
res<-evolafit(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,
                # coancestry parameters
                A=A, lambda=c(0.5,0), nQTLperInd = 2,
                # selection parameters
                propSelBetween = 0.5, propSelWithin =0.5,
                nGenerations = 40)

best = which(res$pheno[,1]==max(res$pheno[,1]))[1];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 phenotpic data for 2 traits measured in 1254 single cross hybrids coming from the cross of Flint x Dent heterotic groups. In addition contains the genotipic data (35,478 markers) for each of the 123 Dent lines and 86 Flint lines. The purpose of this data is to demosntrate 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-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_technow)
DT <- DT_technow
DT$occ <- 1; DT$occ[1]=0
M <- M_technow


  A <- A.mat(M)
  # run the genetic algorithm
  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,
                # coancestry parameters
                A=A, lambda=c(0.4,0), nQTLperInd = 70,
                # selection parameters
                propSelBetween = 0.5, propSelWithin =0.5,
                nGenerations = 20)

  best = bestSol(res)[1]
  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 Triticarte 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, $x_{ij} = Bernoulli(\hat{p}_j)$, 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 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.

McLaren, C. G., L. Ramos, C. Lopez, and W. Eusebio. 2000. "Applications of the geneaology manegment 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)


res<-evolafit(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,
                # coancestry parameters
                A=As, lambda=c(1,0), nQTLperInd = 80,
                # selection parameters
                propSelBetween = 0.5, propSelWithin =0.5,
                nGenerations = 30, verbose = TRUE)


best <- bestSol(res)[1]
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 in the problem replace the genes. Then evolutionary forces are applied to find a close-to-optimal solution.

Usage

```
evolafit(formula, dt,
     constraintsUB, constraintsLB, traitWeight,
     nCrosses=50, nProgeny=40,nGenerations=30, recombGens=1,
     nQTLperInd=NULL, A=NULL, lambda=NULL,
     propSelBetween=1,propSelWithin=0.5,
     fitnessf=NULL, verbose=TRUE, dateWarning=TRUE)
```

Arguments

`formula`	Formula of the form y~x where 'y' refers to the traits or features involved in selecting or putting constraints to the solutions, and 'x'refers to the classifiers of the problem to define the chromosome regions.
`dt`	A dataset containing the features and classifiers.
`constraintsUB`	A numeric vector specifying the upper bound constraints in the traits/features (y). The length is equal to the number of traits/features. If missing is assume infinite for all traits.
`constraintsLB`	A numeric vector specifying the lower bound constraints in the traits/features (y). The length is equal to the number of traits/features. If missing is assume minus infinite for all traits.

traitWeight	A numeric vector specifying the weights that each trait has in the final selection index. The length is equal to the number of traits/features. If missing is assumed equal weight 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 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.
nQTLperInd	The number of levels corresponding to the classifier (x) that should show up as present at the begginning of the simulation. If not specified it will be equal to the number of rows in the dataset/5. See details section.
A	A relationship matrix between the levels of the classifier variable (x).
lambda	A numeric value indicating the weight assigned to the relationship between levels of the classifiers with respect to the selection index. If not specified is assumed to be 0.
propSelBetween	A numeric value between 0 and 1 indicating the proportion of families/crosses that should be selected.
propSelWithin	A numeric value between 0 and 1 indicating the proportion of individuals within families/crosses that should be selected.
fitnessf	An alternative named list defining the fitness function for a given trait. If NULL the default function will be xa - (lam * x'Ax) where x is the contribution vector to the solution, a are the original values assigned to the trait-merit, and A is the covariance between solutions.
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.

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 combinations present for the classifier/genes is controlled by the nQTLperInd parameter. This of course will increase if has an effect in the fitness. The drift force can be controlled by the recombGens parameter.

Value

$M the matrix of haplotypes/solutions after selection.

$score a matrix with scores for different metrics () across generations of evolution.

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

indivPerformance the matrix of x'a, x'Ax, deltaC, nQTLs per solution per generation.

pop AlphaSimR object used for the evolutionary algorithm.

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.
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
)

best = bestSol(res0)[2]
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)
```

`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 desribed 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 *plot the change of values across iterations*

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

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, 14
pmonitor, 15

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, *13*, *15, 16*

GT_wheat *(*DT_wheat*)*, 9

M_technow *(*DT_technow*)*, 7

pareto, 14
plot, *15*
pmonitor, 15

varM, 16