Package 'AlphaPart'

October 22, 2021

Title Partition/Decomposition of Breeding Values by Paths of Information

Description A software that implements a method for partitioning genetic trends to quantify the sources of genetic gain in breeding programmes. The partitioning method is described in Garcia-Cortes et al. (2008) <doi:10.1017/S175173110800205X>. The package includes the main function AlphaPart for partitioning breeding values and auxiliary functions for manipulating data and summarizing, visualizing, and saving results.

Maintainer Gregor Gorjanc <highlander.research.lab@gmail.com>

License GPL (>= 2)

LazyLoad yes

Imports directlabels (>= 1.1), gdata (>= 2.6.0), ggplot2 (>= 0.8.9), pedigree (>= 1.3.1), Rcpp (>= 0.9.4), reshape

Suggests RColorBrewer (>= 1.0-2), truncnorm (>= 1.0-5), knitr, rmarkdown, testthat (>= 3.0.0), covr

LinkingTo Rcpp

Version 0.8.4

Date 2021-10-15

NeedsCompilation yes

VignetteBuilder knitr

RoxygenNote 7.1.2

Encoding UTF-8

LazyData true

Config/testthat/edition 3

Author Gregor Gorjanc [aut, cre] (<https://orcid.org/0000-0001-8008-2787>), Jana Obsteter [aut] (<https://orcid.org/0000-0003-1511-3916>), Thiago de Paula Oliveira [aut] (<https://orcid.org/0000-0002-4555-2584>)

Repository CRAN

Date/Publication 2021-10-22 08:50:15 UTC

R topics documented:

`AlphaPart` *AlphaPart.R*

Description

A function to partition breeding values by a path variable. The partition method is described in García-Cortés et al., 2008: Partition of the genetic trend to validate multiple selection decisions. Animal : an international journal of animal bioscience. DOI: doi: 10.1017/S175173110800205X

Usage

```
AlphaPart(x, pathNA, recode, unknown, sort, verbose, profile,
  printProfile, pedType, colId, colFid, colMid, colPath, colBV,
  colBy, center, centerEBV)
```

Arguments

`x`	data.frame , with (at least) the following columns: individual, father, and mother identif ication, and year of birth; see arguments `colId`, `colFid`, `colMid`, `colPath`, and `colBV`; see also details about the validity of pedigree.
`pathNA`	Logical, set dummy path (to "XXX") where path information is unknown (missing).
`recode`	Logical, internally recode individual, father and, mother identification to `1:n` codes, while missing parents are defined with `0`; this option must be used if identif ications in `x` are not already given as `1:n` codes, see also argument `sort`.
`unknown`	Value(s) used for representing unknown (missing) parent in `x`; this options has an effect only when `recode=FALSE` as it is only needed in that situation.

sort	Logical, initially sort x using orderPed() so that children follow parents in order to make imputation as optimal as possible (imputation is performed within a loop from the first to the last unknown birth year); at the end original order is restored.
verbose	Numeric, print additional information: 0 - print nothing, 1 - print some summaries about the data.
profile	Logical, collect timings and size of objects.
printProfile	Character, print profile info on the fly ("fly") or at the end ("end").
pedType	Character, pedigree type: the most common form is "IPP" for Individual, Parent 1 (say father), and Parent 2 (say mother) data; the second form is "IPG" for Individual, Parent 1 (say father), and one of Grandparents of Parent 2 (say maternal grandfather).
colId	Numeric or character, position or name of a column holding individual identif ication.
colFid	Numeric or character, position or name of a column holding father identif ication.
colMid	Numeric or character, position or name of a column holding mother identif ication or maternal grandparent identif ication if pedType="IPG" .
colPath	Numeric or character, position or name of a column holding path information.
colBV	Numeric or character, position(s) or name(s) of column(s) holding breeding Values.
colBy	Numeric or character, position or name of a column holding group information (see details).
center	Logical, if center=TRUE detect a shift in base population mean and attributes it as parent average effect rather than mendelian sampling effect, otherwise if center=FALSE, the base population values are only accounted as mendelian sampling effect. Default is center = TRUE.
centerEBV	Logical, if centerEBV=TRUE center the EBVs in order to the base population has mean of zero. Default is center = FALSE.

Details

Pedigree in x must be valid in a sense that there are:

- no directed loops (the simplest example is that the individual identification is equal to the identification of a father or mother)
- no bisexuality, e.g., fathers most not appear as mothers
- father and/or mother can be unknown (missing) - defined with any "code" that is dif ferent from existing identif ications

Unknown (missing) values for breeding values are propagated down the pedigree to provide all available values from genetic evaluation. Another option is to cut pedigree links - set parents to unknown and remove them from pedigree prior to using this function - see pedSetBase function. Warning is issued in the case of unknown (missing) values.

In animal breeding/genetics literature the model with the underlying pedigree type "IPP" is often called animal model, while the model for pedigree type "IPG" is often called sire - maternal grandsire model. With a combination of colFid and colMid mother - paternal grandsire model can be accomodated as well.

Argument colBy can be used to directly perform a summary analysis by group, i.e., summary(AlphaPart(...),by="group") See summary.AlphaPart for more. This can save some CPU time by skipping intermediate steps. However, only means can be obtained, while summary method gives more flexibility.

Value

An object of class AlphaPart, which can be used in further analyses - there is a handy summary method (summary.AlphaPart works on objects of AlphaPart class) and a plot method for its output (plot.summaryAlphaPart works on objects of summaryAlphaPart class). Class AlphaPart is a list. The first length(colBV) components (one for each trait and named with trait label, say trt) are data frames. Each data.frame contains:

x	columns from initial data x
trt_pa	parent average
trt_w	Mendelian sampling term
trt_path1, trt_path2, ...	breeding value partitions

The last component of returned object is also a list named info with the following components holding meta information about the analysis:

path	column name holding path information
nP	number of paths
lP	path labels
nT	number of traits
lT	trait labels
warn	potential warning messages associated with this object

If colBy!=NULL the resulting object is of a class summaryAlphaPart, see summary.AlphaPart for details.

If profile=TRUE, profiling info is printed on screen to spot any computational bottlenecks.

References

Garcia-Cortes, L. A. et al. (2008) Partition of the genetic trend to validate multiple selection decisions. Animal, 2(6):821-824. doi: 10.1017/S175173110800205X

See Also

summary.AlphaPart for summary method that works on output of AlphaPart, pedSetBase for setting base population, pedFixBirthYear for imputing unknown (missing) birth years, orderPed in **pedigree** package for sorting pedigree

Examples

```
## Small pedigree with additive genetic (=breeding) values
ped <- data.frame(  id=c(  1,   2,   3,   4,   5,   6),
                   fid=c(  0,   0,   2,   0,   4,   0),
                   mid=c(  0,   0,   1,   0,   3,   3),
                   loc=c("A", "B", "A", "B", "A", "A"),
                   gen=c(  1,   1,   2,   2,   3,   3),
                  trt1=c(100, 120, 115, 130, 125, 125),
                  trt2=c(100, 110, 105, 100,  85, 110))

## Partition additive genetic values
tmp <- AlphaPart(x=ped, colBV=c("trt1", "trt2"))
print(tmp)

## Summarize by generation
summary(tmp, by="gen")


## There are also two demos
  demo(topic="AlphaPart_deterministic", package="AlphaPart",
       ask=interactive())
  demo(topic="AlphaPart_stochastic",     package="AlphaPart",
       ask=interactive())
```

`AlphaPart.ped` *Sample pedigree for partition.*

Description

A dataset containing pedigree information and breeding values for six individuals.

Usage

```
AlphaPart.ped
```

Format

A data frame with 6 rows and 8 variables:

IId individual's ID

FId Father's ID

MId Mother's ID

gen Generation

country Country

gender Individual's sex

bv1 Breeding value for trait 1

bv2 Breeding value for trait 1

Source

Simulation.

`AlphaPartSubset` *AlphaPartSubset.R*

Description

A function to choose the partition paths to keep.

Usage

```
AlphaPartSubset(x, paths = NULL)
```

Arguments

`x`	AlphaPart or summaryAlphaPart, object from the `AlphaPart(...)` or `summary(AlphaPart(...),...)` call.
`paths`	Character, names of paths to be kept.

Details

Displaying results of partitions for many paths is often confusing. This function helps in selecting only paths of interest. Unspecified paths are removed from the input object `x`. Meta information is modified accordingly. Default setting does nothing.

Value

An object of class `AlphaPart` or `summaryAlphaPart` with only some paths. Meta information in slot "info" is modified as well.

See Also

`AlphaPart` for the main method, `summary.AlphaPart` for summary method that works on output of `AlphaPart`, `AlphaPartSum` for sum method.

Examples

```
## Small pedigree with additive genetic (=breeding) values
ped <- data.frame(  id=c(  1,   2,   3,   4,   5,   6),
                   fid=c(  0,   0,   2,   0,   4,   0),
                   mid=c(  0,   0,   1,   0,   3,   3),
                   loc=c("A", "B", "A", "B", "A", "A"),
                   gen=c(  1,   1,   2,   2,   3,   3),
                  trt1=c(100, 120, 115, 130, 125, 125),
                  trt2=c(100, 110, 105, 100,  85, 110))

## Partition additive genetic values
```

```
(tmp <- AlphaPart(x=ped, colBV=c("trt1", "trt2")))

## Keep some partitions (working on object of class AlphaPart)
(tmp2 <- AlphaPartSubset(x=tmp, paths="A"))

## Summarize by generation
(tmpS <- summary(tmp, by="gen"))

## Keep some partitions (working on object of class summaryAlphaPart)
(tmpS2 <- AlphaPartSubset(x=tmpS, paths="A"))

## ... must be equal to
(tmpS3 <- summary(tmp2, by="gen"))
```

AlphaPartSum *AlphaPartSum.R*

Description

A function to sum partitions of several paths.

Usage

```
AlphaPartSum(
  x,
  map = NULL,
  remove = TRUE,
  zeroPath = TRUE,
  call = "AlphaPartSum"
)
```

Arguments

x	summaryAlphaPart, object from the `AlphaPart(...)` or `summary(AlphaPart(...),...)` call.
map	List, a map of summing paths; see details and examples.
remove	Logical, remove original paths or not.
zeroPath	Logical, set called path to zero if it does not exist.
call	character, for internal use with `AlphaPartSubset`).

Details

Sometimes partitions of particular paths are very small or we want to sum paths that have some similarity. These actions are easy to achive manually but this functions provides a way to do this consistently with the given object `x`.

Arguments `map` must be a list of vectors of length at least two. Vectors of length one are skipped. The idea is that the first element is the new or existing path into which we add up all the remaining

specified paths, say `list(c("A","B"),c("X","X","Y"),c("Z","X"))` would imply A = B, X = X + Y, and Z = X = X + Y. Note that once X is changed its changed value is used in further calculations. Specify different (new) names for new targets if you want to avoid this.

Be carefull with `remove=TRUE`, which is the default setting, as all partitions defined after the first (target/new) partition in vector in list will be removed, for example with `list(c("A","B"),c("X","X","Y"),c("Z","X"))` partitions B and Y will be removed, while X will not be removed as it is defined as a target/new partition.

Value

An object of class `AlphaPart` or `summaryAlphaPart` with modified partitions. Meta information in slot "info" is modified as well.

See Also

`AlphaPart` for the main method, `summary.AlphaPart` for summary method that works on output of `AlphaPart`, `AlphaPartSubset` for subset/keep method

Examples

```
## Small pedigree with additive genetic (=breeding) values
ped <- data.frame(  id=c(  1,   2,   3,   4,   5,   6),
                   fid=c(  0,   0,   2,   0,   4,   0),
                   mid=c(  0,   0,   1,   0,   3,   3),
                   loc=c("A", "B", "A", "B", "A", "A"),
                   gen=c(  1,   1,   2,   2,   3,   3),
                  trt1=c(100, 120, 115, 130, 125, 125),
                  trt2=c(100, 110, 105,  140,  85, 110))

## Partition additive genetic values
(tmp <- AlphaPart(x=ped, colBV=c("trt1", "trt2")))

## Sum some partitions (working on object of class AlphaPart)
(tmp2 <- AlphaPartSum(x=tmp, map=list(c("X", "A", "B"), c("A", "B"))))

## Summarize by generation
(tmpS <- summary(tmp, by="gen"))

## Sum some partitions (working on object of class summaryAlphaPart)
(tmpS2 <- AlphaPartSum(x=tmpS, map=list(c("X", "A", "B"), c("A", "B"))))

## ... must be equal to
(tmpS3 <- summary(tmp2, by="gen"))
```

`pedFixBirthYear` *pedFixBirthYear.R*

Description

A function to fix (impute) missing birth years in pedigree.

Usage

```
pedFixBirthYear(
  x,
  interval,
  down = FALSE,
  na.rm = TRUE,
  sort = TRUE,
  direct = TRUE,
  report = TRUE,
  colId = 1,
  colFid = 2,
  colMid = 3,
  colBY = 4
)
```

Arguments

x	data.frame , with (at least) the following columns: individual, father, and mother identification, and year of birth; see arguments colId, colFid, colMid, and colBY
interval	Numeric, a value for generation interval in years.
down	Logical, the default is to impute birth years based on the birth year of children starting from the youngest to the oldest individuals, while with down=TRUE birth year is imputed based on the birth year of parents in the opposite order.
na.rm	Logical, remove NA values when searching for the minimal (maximal) year of birth in children (parents); setting this to FALSE can lead to decreased success of imputation
sort	Logical, initially sort x using orderPed() so that children follow parents in order to make imputation as optimal as possible (imputation is performed within a loop from the first to the last unknown birth year); at the end original order is restored.
direct	Logical, insert inferred birth years immediately so they can be used for successive individuals within the loop.
report	Logical, report success.
colId	Numeric or character, position or name of a column holding individual identification.
colFid	Numeric or character, position or name of a column holding father identification.
colMid	Numeric or character, position or name of a column holding mother identification.
colBY	Numeric or character, position or name of a column holding birth year.

Details

Warnings are issued when there is no information to use to impute birth years or missing values (NA) are propagated.

Arguments down and na.rm allow for repeated use of this function, i.e., with down=FALSE and with down=TRUE (both in combination with na.rm=TRUE) in order to propagate information over the pedigree until "convergence".

This function can be very slow on large pedigrees with extensive missingness of birth years.

Value

Object x with imputed birth years based on the birth year of children or parents. If report=TRUE success is printed on the screen as the number of initially, fixed, and left unknown birth years is printed.

See Also

orderPed in **pedigree** package

Examples

```
## Example pedigree with missing (unknown) birth year for some individuals
ped0 <- data.frame(     id=c( 1, 2, 3,  4, 5, 6, 7,  8, 9, 10, 11, 12, 13, 14),
                        fid=c( 0, 0, 0,  1, 1, 1, 3,  3, 3,  5,  4,  0,  0, 12),
                        mid=c( 0, 0, 0,  2, 0, 2, 2,  2, 5,  0,  0,  0,  0, 13),
                        birth_dt=c(NA, 0, 1, NA, 3, 3, 3, 3, 4, 4, 5, NA, 6, 6) + 2000)

## First run - using information from children
ped1 <- pedFixBirthYear(x=ped0, interval=1)

## Second run - using information from parents
ped2 <- pedFixBirthYear(x=ped1, interval=1, down=TRUE)

## Third run - using information from children, but with no success
ped3 <- pedFixBirthYear(x=ped2, interval=1)
```

---

pedSetBase *pedSetBase.R*

Description

A function to set the base population in the pedigree.

Usage

```
pedSetBase(
  x,
  keep = NULL,
  unknown = NA,
  report = TRUE,
  colId = 1,
  colFid = 2,
  colMid = 3
)
```

Arguments

x	data.frame , with (at least) the following columns: individual, father, and mother identification, and year of birth; see arguments colId, colFid, colMid, and colBY
keep	Logical, indicator that defines which individuals should stay in the the pedigree; see details.
unknown	Value used to represent unknown/missing identification
report	Logical, report success.
colId	Numeric or character, position or name of a column holding individual identification.
colFid	Numeric or character, position or name of a column holding father identification.
colMid	Numeric or character, position or name of a column holding mother identification.

Details

Base population in the pedigree is set by removing rows for some individuals, while their presence as parents is also removed.

Arguments down and na.rm allow for repeated use of this function, i.e., with down=FALSE and with down=TRUE (both in combination with na.rm=TRUE) in order to propagate information over the pedigree until "convergence".

This function can be very slow on large pedigrees with extensive missingness of birth years.

Value

Object x with removed rows for some individuals and their presence as parents. If report=TRUE progress is printed on the screen.

See Also

orderPed in **pedigree** package

Examples

```
## Example pedigree
ped <- data.frame(      id=1:10,
                       fid=c(0, 0, 0, 1, 1, 1, 3, 3, 3, 5),
                       mid=c(0, 0, 0, 2, 0, 2, 2, 2, 5, 0),
                  birth_dt=c(0, 0, 1, 2, 3, 3, 3, 4, 4, 5) + 2000)

## Set base population as those individuals that were born after year 2002
pedSetBase(x=ped, keep=ped$birth_dt > 2002, unknown=0)
```

`plot.summaryAlphaPart` *A function to plot summary of partitioned breeding values.*

Description

A function to plot summary of partitioned breeding values.

Usage

```
## S3 method for class 'summaryAlphaPart'
plot(x, by, sortValue,
  sortValueFUN, sortValueDec, addSum, paths, xlab, ylab, xlim, ylim,
  color, lineSize, lineType, lineTypeList, useDirectLabels, method,
  labelPath, ...)
```

Arguments

`x`	summaryAlphaPart, object from the `AlphaPart(...)` or `summary(AlphaPart(...),...)` call.
`by`	Character, the name of a column by which summary function FUN should be applied; if `NULL` (default) summary is given for the whole table.
`sortValue`	Logical, affect legend attributes via sort of paths according to `sortValueFUN` function; if not logical, then ordered paths are given as a character vector.
`sortValueFUN`	Function, that produces single value for one vector, say `mean` or `sum`.
`sortValueDec`	Logical, sort decreasing.
`addSum`	Logical, plot the overall trend.
`paths`	Character or list or characters, name of paths to plot; if `NULL` plot all paths; see examples.
`xlab`	Character, x-axis label.
`ylab`	Character, y-axis label; can be a vector of several labels if there are more traits in `x` (recycled!).
`xlim`	Numeric, a vector of two values with x-axis limits; use a list of vectors for more traits.
`ylim`	Numeric, a vector of two values with y-axis limits; use a list of vectors for more traits.
`color`	Character, color names; by default a set of 54 colors is predefined from the **RColorBrewer** package; in addition a black colour is attached at the begining for the overall trend; if there are more paths than colors then recycling occours.
`lineSize`	Numeric, line width.
`lineType`	Numeric, line type (recycled); can be used only if lineTypeList=NULL.

lineTypeList	List, named list of numeric values that help to point out a set of paths (distinguished with line type) within upper level of paths (distinguished by, color), e.g., lineTypeList=list("-1"=1, "-2"=2, def=1) will lead to use of line type 1 for paths having "-1" at the end of path name and line type 2, for paths having "-2" at the end of path name, while line type 1 (default) will, be used for other paths; specification of this argument also causes recycling of colors for the upper level of paths; if NULL all lines have a standard line type, otherwise lineType does not have any effect.
useDirectLabels	Logical, use directlabels package for legend.
method	List, method for direct.label.
labelPath	Character, legend title; used only if useDirectLabels=FALSE.
...	Arguments passed to other functions (not used at the moment).

Details

Information in summaries of partitions of breeding values can be overhelming due to a large volume of numbers. Plot method can be used to visualise this data in eye pleasing way using ggplot2 graphics.

Value

A list of ggplot objects that can be further modified or displayed. For each trait in x there is one plot visualising summarized values.

Examples

```
## Partition additive genetic values by country
(res <- AlphaPart(x=AlphaPart.ped, colPath="country", colBV=c("bv1", "bv2")))

## Summarize population by generation (=trend)
(ret <- summary(res, by="gen"))

## Plot the partitions
p <- plot(ret, ylab=c("bv for trait 1", "bv for trait 2"), xlab="Generation")
print(p[[1]]$abs)
print(p[[2]]$abs)
print(p)

## Partition additive genetic values by country and sex
AlphaPart.ped$country.gender <- with(AlphaPart.ped, paste(country, gender, sep="-"))
(res <- AlphaPart(x=AlphaPart.ped, colPath="country.gender", colBV=c("bv1", "bv2")))

## Summarize population by generation (=trend)
(ret <- summary(res, by="gen"))

## Plot the partitions
p <- plot(ret, ylab=c("BV for trait 1", "BV for trait 2"), xlab="Generation")
```

```
print(p)
p <- plot(ret, ylab=c("BV for trait 1", "BV for trait 2"), xlab="Generation",
        lineTypeList=list("-1"=1, "-2"=2, def=3))
print(p)
p <- plot(ret, ylab=c("BV for trait 1", "BV for trait 2"), xlab="Generation",
      lineTypeList=list("-1"=1, "-2"=2, def=3), useGgplot2=FALSE, useDirectLabels = FALSE)
print(p)

## Plot control (color and type of lines + limits)
p <- plot(ret, ylab=c("BV for trait 1", "BV for trait 2"), xlab="Generation",
        useGgplot2=TRUE, color=c("green", "gray"), lineType=c(2, 3),
        sortValue=FALSE, lineSize=4,
        xlim=c(-1, 7))
print(p)
```

`print.AlphaPart` *Print method for the output of AlphaPart function.*

Description

Partitioning of breeding values if often performed on quite large datasets, which quickly fills in the whole screen. Print method therefore prints out paths, number of individuals and the first and the last few lines for each trait to quickly see what kind of data is in `x`.

Usage

```
## S3 method for class 'AlphaPart'
print(x, n, ...)
```

Arguments

`x`	AlphaPart, output object from `AlphaPart` function.
`n`	Integer, number of the first and last rows in `x` to print out using `head` and `tail`.
`...`	Arguments passed to `print` function.

See Also

`AlphaPart`, `head`, `tail`.

Examples

```
## Small pedigree with additive genetic (=breeding) values
ped <- data.frame(  id=c(  1,   2,   3,   4,   5,   6),
                    fid=c(  0,   0,   2,   0,   4,   0),
                    mid=c(  0,   0,   1,   0,   3,   3),
                    loc=c("A", "B", "A", "B", "A", "A"),
                    gen=c(  1,   1,   2,   2,   3,   3),
                   trt1=c(100, 120, 115, 130, 125, 125),
```

```
                  trt2=c(100, 110, 105, 100,  85, 110))

## Partition additive genetic values
tmp <- AlphaPart(x=ped, colBV=c("trt1", "trt2"))
print(tmp)

## Summarize by generation
summary(tmp, by="gen")


## There are also two demos
  demo(topic="AlphaPart_deterministic", package="AlphaPart",
       ask=interactive())
  demo(topic="AlphaPart_stochastic",     package="AlphaPart",
       ask=interactive())
```

`print.plotSummaryAlphaPart` *Print a plot generate by the function* `plotSummaryAlphaPart`

Description

Plot output object from `plot.summaryAlphaPart`.

Usage

```
## S3 method for class 'plotSummaryAlphaPart'
print(x, ask, ...)
```

Arguments

`x`	plotSummaryAlphaPart, output object from `plot.summaryAlphaPart` function
`ask`	Logical, ask before printing another plot?
`...`	Arguments passed to other functions (not used at the moment).

See Also

`plot.summaryAlphaPart`

Examples

```
## Partition additive genetic values
(res <- AlphaPart(x=AlphaPart.ped, colPath="country", colBV=c("bv1", "bv2")))

## Summarize population by generation (=trend)
(ret <- summary(res, by="gen"))

## Plot the partitions
```

```
p <- plot(ret, ylab=c("BV for trait 1", "BV for trait 2"), xlab="Generation")
print(p[[1]])
print(p[[2]])
#print(p)
```

`print.summaryAlphaPart` *Print method for objects of the class summaryAlphaPart.*

Description

Print method for objects of the class `summaryAlphaPart` (result of `summary(AlphaPart(...))`).

Usage

```
## S3 method for class 'summaryAlphaPart'
print(x, ...)
```

Arguments

`x`	summaryAlphaPart, output object from `summary.AlphaPart` function.
`...`	Arguments passed to other functions (not used at the moment).

See Also

`summary.AlphaPart`

Examples

```
## --- Partition additive genetic values by loc ---
res <- AlphaPart(x=AlphaPart.ped, colPath="country", colBV=c("bv1", "bv2"))

## Summarize whole population
ret <- summary(res)

## Summarize population by generation (=trend)
ret <- summary(res, by="gen")

## Summarize population by generation (=trend) but only for domestic location
ret <- summary(res, by="gen", subset=res[[1]]$country == "domestic")

## --- Partition additive genetic values by loc and gender ---

AlphaPart.ped$country.gender <- with(AlphaPart.ped, paste(country, gender, sep="-"))
res <- AlphaPart(x=AlphaPart.ped, colPath="country.gender", colBV=c("bv1", "bv2"))

## Summarize population by generation (=trend)
ret <- summary(res, by="gen")

## Summarize population by generation (=trend) but only for domestic location
ret <- summary(res, by="gen", subset=res[[1]]$country == "domestic")
```

`savePlot` *Save plot method for* `AlphaPart`

Description

Save plot method for `AlphaPart`

Usage

```
savePlot(...)
```

Arguments

`...`	Arguments passed to `type` specific methods, say `width` and `height` for `type="pdf"` etc.

Value

Beside the side effect of saving plots to disk, filenames are printed on screen during the process and at the end invisibly returned.

`savePlot.plotSummaryAlphaPart` *Save plot objects on the disk for permanent storage. Function* `savePlot` *from the* **grDevices** *package works for current page on graphical device. This is an attempt to make this function generic so that one can define* `savePlot` *methods for particular needs.*

Description

Save plot objects of class `plotSummaryAlphaPart` on the disk for permanent storage.

Usage

```
## S3 method for class 'plotSummaryAlphaPart'
savePlot(x, filename, type,
  device, pre.hook, traitsAsDir, ...)

## Default S3 method:
savePlot(...)
```

Arguments

x	Object on which to chose savePLot method.
filename	Character, filename to save to.
type	Character, file/device type.
device	Device, the device to save from.
pre.hook	Function, call some code before calling print method for plots (see examples).
traitsAsDir	Logical, should plots be saved within trait folders; the construction is file.path(dirname(file),trait folders are created if they do not exist.
...	Arguments passed to type specific methods, say width and height for type="pdf" etc.

Value

Beside the side effect of saving plots to disk, filenames are printed on screen during the process and at the end invisibly returned.

See Also

savePlot help page on the default savePlot method in the **grDevices** package; savePlot.plotSummaryAlphaPart help page on the method for the objects of plotSummaryAlphaPart class; and plot.summaryAlphaPart for ploting results of summaryAlphaPart object.

Examples

```
## Partition additive genetic values
res <- AlphaPart(x=AlphaPart.ped, colPath="country", colBV=c("bv1", "bv2"))

## Summarize population by generation (=trend)
ret <- summary(res, by="gen")

## Plot the partitions
p <- plot(ret, ylab=c("BV for trait 1", "BV for trait 2"), xlab="Generation")

## Save the plots
tmp <- savePlot(x = p, filename="test", type="png")

## Remove the files
unlink(tmp)
```

---

summary.AlphaPart *A function to summarize AlphaPart object.*

Description

Breedng values of individuals are often summarized, either by year of birth or some other classification. Function summary.AlphaPart provides a way to ease the computation of such summaries on partitions of breeding values.

Usage

```
## S3 method for class 'AlphaPart'
summary(object, by, FUN, labelSum, subset,
  sums, ...)
```

Arguments

object	AlphaPart, output object from AlphaPart function.
by	Character, the name of a column by which summary function FUN should be applied; if NULL (default) summary is given for the whole table.
FUN	Function, which function should be used in summary; function should return single value per each level of by.
labelSum	Character, label used for the overall breeding value.
subset	Logical, perform summary only on a subset of object subsetted by this argument.
sums	Logical, link between AlphaPart and summary.AlphaPart() (only for internal use!).
...	Arguments passed to other functions (not used at the moment).

Value

An object of class summaryAlphaPart, which is a list of data frames with summary statistics on breeding value partitions. For each trait there a dataframe holds summary for the "whole/original" breeding value and its partitions. In addition another list is added (named info) with the following components holdinfg meta info:

path	column name holding path information
nP	number of paths
lP	path labels
nT	number of traits
lT	trait labels
by	column name of variable by which summary was performed
warn	potential warning messages associated with this object
labelSum	column name of summary for "whole/original" breeding values

There is a handy plot method (plot.summaryAlphaPart) for output.

See Also

AlphaPart for partitioning breeding values, plot.summaryAlphaPart for plotting output of summary method

Examples

```
## --- Partition additive genetic values by loc ---
res <- AlphaPart(x=AlphaPart.ped, colPath="country", colBV=c("bv1", "bv2"))

## Summarize whole population
ret <- summary(res)

## Summarize population by generation (=trend)
ret <- summary(res, by="gen")

## Summarize population by generation (=trend) but only for domestic location
ret <- summary(res, by="gen", subset=res[[1]]$country == "domestic")

## --- Partition additive genetic values by loc and gender ---

AlphaPart.ped$country.gender <- with(AlphaPart.ped, paste(country, gender, sep="-"))
res <- AlphaPart(x=AlphaPart.ped, colPath="country.gender", colBV=c("bv1", "bv2"))

## Summarize population by generation (=trend)
ret <- summary(res, by="gen")

## Summarize population by generation (=trend) but only for domestic location
ret <- summary(res, by="gen", subset=res[[1]]$country == "domestic")
```

`write.csv` *write.csv.R*

Description

Save summaries of partitioned breeding values to CSV files on disk for further analyses of processing with other software or just for saving (backing up) results.

Usage

```
write.csv(...)

## Default S3 method:
write.csv(...)

## S3 method for class 'AlphaPart'
write.csv(x, file, traitsAsDir = FALSE, csv2 = TRUE, row.names = FALSE, ...)

## S3 method for class 'summaryAlphaPart'
write.csv(x, file, traitsAsDir = FALSE, csv2 = TRUE, row.names = FALSE, ...)
```

Arguments

`...`	Other options passed to `write.csv2` or `write.csv`.

x	AlphaPart, object returned from AlphaPart function or summaryAlphaPart, object returned from summary.AlphaPart function.
file	Character, file name with or without .csv extension, e.g., both "file" and "file.csv" are valid.
traitsAsDir	Logical, should results be saved within trait folders; the construction is file.path(dirname(file),trai folders are created if they do not exist.
csv2	Logical, export using write.csv2 or write.csv.
row.names	Logical, export row names as well?

Details

Function write.csv from the **utils** package works when exported object is a data.frame or a matrix. This is an attempt to make this function generic so that one can define write.csv methods for other objects.

Value

write.csv
: See write.csv for details.

write.csv.AlphaPart
: For each trait (list component in x) a file is saved on disk with name "AlphaPart_trait.csv", where the file will hold original data and breeding value partitions. With traitsAsDir=TRUE files are saved as "trait/file_trait.csv". File names are printed on screen during the process of export and at the end invisibly returned.

litemwrite.csv.summaryAlphaPartFor each trait (list component in x) a file partitions named With traitsAsDir=TRUE files are saved as "trait/file_trait_*.csv". File names are printed on screen during the process of export and at the end invisibly returned.

Methods (by class)

- default: Default write.csv method.
- AlphaPart: Save partitioned breeding values to CSV files on disk on disk for further analyses or processing with other software or just for saving (backing up) results.
- summaryAlphaPart: Save summaries of partitioned breeding values to CSV files on disk for further analyses of processing with other software or just for saving (backing up) results.

See Also

write.csv help page on the default write.csv and write.csv2 methods in the **utils** package; summary.AlphaPart and AlphaPart help pages on the objects of summaryAlphaPart and AlphaPart classes.

Examples

```
## Partition additive genetic values
res <- AlphaPart(x=AlphaPart.ped, colPath="country", colBV=c("bv1", "bv2"))

## Write summary on the disk and collect saved file names
```

```
fileName <- file.path(tempdir(), "AlphaPart")
ret <- write.csv(x=res, file=fileName)
print(ret)
file.show(ret[1])

## Clean up
files <- dir(path=tempdir(), pattern="AlphaPart*")
unlink(x=files)
```

Index