Package ‘optiSolve’

October 14, 2022

Type Package

Title Linear, Quadratic, and Rational Optimization

Version 1.0

Date 2021-10-13

Author Robin Wellmann

Maintainer Robin Wellmann <r.wellmann@uni-hohenheim.de>

Depends R (>= 3.4)

Description Solver for linear, quadratic, and rational programs with linear, quadratic, and rational constraints. A unified interface to different R packages is provided. Optimization problems are transformed into equivalent formulations and solved by the respective package. For example, quadratic programming problems with linear, quadratic and rational constraints can be solved by augmented Lagrangian minimization using package 'alabama', or by sequential quadratic programming using solver 'slsqp'. Alternatively, they can be reformulated as optimization problems with second order cone constraints and solved with package 'cccp'.

License GPL-2

Imports Matrix, shapes, alabama, cccp, nloptr, MASS, methods, plyr, stringr, stats, Rcpp (>= 0.12.4)

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2021-10-13 12:32:04 UTC

R topics documented:

`optiSolve-package` *Linear, Quadratic, and Rational Optimization*

Description

Solver for linear, quadratic, and rational programs with linear, quadratic, and rational constraints. A unified interface to different R packages is provided. Optimization problems are transformed into equivalent formulations and solved by the respective package. For example, quadratic programming problems with linear, quadratic and rational constraints can be solved by augmented Lagrangian minimization using package 'alabama', or by sequential quadratic programming using solver 'slsqp'. Alternatively, they can be reformulated as optimization problems with second order cone constraints and solved with package 'cccp'.

Details

The following steps are included in solving a constrained optimization problem (cop):

1) Define the objective with one of the following functions:

linfun	defines a linear objective function,
quadfun	defines a quadratic objective function,
ratiofun	defines a rational objective function.

2) Define the constraints by using the following functions:

lincon	defines linear equality and inequality constraints,
quadcon	defines quadratic constraints,
ratiocon	defines rational constraints,
lbcon	defines lower bounds for the variables,
ubcon	defines upper bounds for the variables.

3) Put the objective function and the constraints together to define the optimization problem:

cop   defines a constrained optimization problem.

4) Solve the optimization problem:

solvecop   solves a constrained optimization problem.

5) Check if the solution fulfils all constraints:

validate   checks if the solution fulfils all constraints, and calculates the values of the constraints.

Author(s)

Robin Wellmann

Maintainer: Robin Wellmann <r.wellmann@uni-hohenheim.de>

References

Kraft, D. (1988). A software package for sequential quadratic programming, Technical Report DFVLR-FB 88-28, Institut fuer Dynamik der Flugsysteme, Oberpfaffenhofen, July 1988.

Lange K, Optimization, 2004, Springer.

Madsen K, Nielsen HB, Tingleff O, Optimization With Constraints, 2004, IMM, Technical University of Denmark.

`adjust` *Adjust Constraints and Objective Functions*

Description

Constraints and objective functions are adjusted so that they refer to a larger or smaller set of variables.

Usage

```
adjust(x, ids)
```

Arguments

`x`	Constraint or objective function of class `"linFun"`, `"linCon"`, `"quadFun"`, `"quadCon"`, `"ratioFun"`, and `"ratioCon"`.
`ids`	Vector with ids of the variables.

Details

Constraints and objective functions are adjusted so that they refer to a larger or smaller set of variables. Additional variables do not affect the value of the constraint or objective function.

Value

A data frame (invisible) containing values and bounds of the constraints, the value of the objective function, and column `valid` which is TRUE if all constraints are fulfilled.

See Also

The main function for solving constrained programming problems is solvecop.

---

cop *Constrained Optimization Problem*

Description

Define a constrained optimization problem with a linear, quadratic, or rational objective function, and linear, quadratic, rational, and boundary constraints.

Usage

```
cop(f, max=FALSE, lb=NULL, ub=NULL, lc=NULL, ...)
```

Arguments

`f`	Objective function, defined with function linfun, quadfun, or ratiofun.
`max`	Logical value. Should the function be maximized? This is possible only for linear objective functions.
`lb`	Lower bounds for the variables, defined with function lbcon.
`ub`	Upper bounds for the variables, defined with function ubcon.
`lc`	Linear inequality and equality constraints, defined with function lincon.
`...`	Quadratic and rational inequality constraints, defined with functions quadcon and ratiocon.

Details

Define a constrained optimization problem with a linear, quadratic, or rational objective function, and linear, quadratic, rational, and boundary constraints. The optimization problem can be solved with function solvecop.

Value

An object of class `COP`, which may contain the following components

`f`	List with S3-class "linFun", "quadFun", or "ratioFun", defining the objective function
`max`	Logical value. Should the objective function be maximized?
`lb`	List with S3-class "lbCon", defining lower bounds.
`ub`	List with S3-class "ubCon", defining upper bounds.
`lc`	List with S3-class "linCon", defining linear constraints
`qc`	List with S3-class "quadCon", defining quadratic constraints
`rc`	List with S3-class "ratioCon", defining rational constraints
`x`	Vector with NAs
`id`	Vector with names of the variables that are to be optimized
`madeDefinite`	Logical variable indicating whether non-positive-semidefinite matrices have already been approximated by positive-definite matrices.

Author(s)

Robin Wellmann

See Also

The main function for solving constrained programming problems is solvecop.

`lbcon` *Lower Bounds*

Description

Define lower bounds for the variables of the form

$$val <= x.$$

Usage

```
lbcon(val=numeric(0), id=seq_along(val))
```

Arguments

`val`	Numeric vector with lower bounds for the variables. If `val` is a single value, then this value will be used for all variables in vector `id`.
`id`	Vector defining the names of the variables to which the constraint applies. Each variable name corresponds to one component of `x`. Variable names must be consistent across constraints.

Details

Define lower bounds for the variables of the form

$$val <= x.$$

Vector x contains only the variables included in argument id.

Value

An object of class lbCon.

See Also

The main function for solving constrained programming problems is solvecop.

Examples

```
### Linear programming with linear and quadratic constraints ###
### Example from animal breeding                              ###
### The mean breeding value BV is maximized whereas the      ###
### mean kinship in the offspring x'Qx+d is restricted       ###
### Lower and upper bounds for females are identical, so     ###
### their contributions are not optimized.                   ###
### Lower and upper bounds for some males are defined.       ###

data(phenotype)
data(myQ)

A   <- t(model.matrix(~Sex-1, data=phenotype))
A[,1:5]
val <- c(0.5,  0.5)
dir <- c("==","==")
Nf  <- sum(phenotype$Sex=="female")
id  <- phenotype$Indiv

lbval <- setNames(rep(0,  length(id)), id)
ubval <- setNames(rep(NA, length(id)), id)
lbval[phenotype$Sex=="female"] <- 1/(2*Nf)
ubval[phenotype$Sex=="female"] <- 1/(2*Nf)
lbval["276000102379430"] <- 0.02
ubval["276000121507437"] <- 0.03

mycop <- cop(f  = linfun(a=phenotype$BV, id=id, name="BV"),
             max= TRUE,
             lb = lbcon(lbval, id=id),
             ub = ubcon(ubval, id=id),
             lc = lincon(A=A, dir=dir, val=val, id=id),
             qc = quadcon(Q=myQ, d=0.001, val=0.045,
                          name="Kinship", id=rownames(myQ)))

res <- solvecop(mycop, solver="cccp2", quiet=FALSE)
```

```
Evaluation <- validate(mycop, res)

#           valid solver  status
#            TRUE  cccp2 optimal
#
#   Variable     Value      Bound    OK?
#   -------------------------------------
#   BV           0.5502 max        :
#   -------------------------------------
#   lower bounds all x  >=  lb     : TRUE
#   upper bounds all x  <=  ub     : TRUE
#   Sexfemale    0.5    ==  0.5    : TRUE
#   Sexmale      0.5    ==  0.5    : TRUE
#   Kinship      0.045  <=  0.045  : TRUE
#   -------------------------------------

res$x["276000102379430"]

res$x["276000121507437"]
```

`lincon` *Linear Constraints*

Description

Define linear equality and inequality constraints of the form

$$Ax + ddirval$$

Usage

```
lincon(A, d=rep(0, nrow(A)), dir=rep("==",nrow(A)), val=rep(0, nrow(A)),
   id=1:ncol(A), use=rep(TRUE,nrow(A)), name=rownames(A))
```

Arguments

`A`	Numeric matrix of the constraint coefficients.
`d`	Numeric vector.
`dir`	Character vector with the directions of the constraints. Each element must be one of "<=", "==", and ">=".
`val`	Numeric vector with threshold values.
`id`	Vector (if present), defining the names of the variables to which the constraint applies. Each variable name corresponds to one component of `x`. Variable names must be consistent across constraints.
`use`	Logical vector indicating the constraints to be included in the optimization problem. If `use[i]=FALSE`, then linear constraint `i` does not affect the result, but the value of the linear function `A[i,] x + d[i]` will be reported by function validate.
`name`	Vector with names of the constraints.

Details

Define linear inequality and equality constraints of the form

$$Ax + ddirval$$

(component wise). If parameter `id` is specified, then vector `x` contains only the indicated variables.

Value

An object of class `linCon`.

See Also

The main function for solving constrained programming problems is `solvecop`.

Examples

```
### Quadratic programming with linear constraints       ###
### Example from animal breeding                        ###
### The mean kinship in the offspring x'Qx+d is minized ###
### and the mean breeding value is restricted.          ###

data(phenotype)
data(myQ)

A   <- t(model.matrix(~Sex+BV-1, data=phenotype))
A[,1:5]
val <- c(0.5, 0.5, 0.40)
dir <- c("==","==",">=")

mycop <- cop(f  = quadfun(Q=myQ, d=0.001, name="Kinship", id=rownames(myQ)),
             lb = lbcon(0,  id=phenotype$Indiv),
             ub = ubcon(NA, id=phenotype$Indiv),
             lc = lincon(A=A, dir=dir, val=val, id=phenotype$Indiv))

res <- solvecop(mycop, solver="cccp", quiet=FALSE)

validate(mycop, res)

#            valid solver  status
#             TRUE   cccp optimal
#
#   Variable     Value      Bound    OK?
#   -------------------------------------
#   Kinship      0.0322 min        :
#   -------------------------------------
#   lower bounds all x  >=  lb     : TRUE
#   Sexfemale    0.5    ==  0.5    : TRUE
#   Sexmale      0.5    ==  0.5    : TRUE
#   BV           0.4    >=  0.4    : TRUE
#   -------------------------------------
```

linfun — *Linear Objective Function*

Description

Define a linear objective function of the form

$$f(x) = a'x + d$$

.

Usage

```
linfun(a, d=0, id=1:length(a), name="lin.fun")
```

Arguments

a	Numeric vector of the coefficients.
d	Numeric value.
id	Vector defining the names of the variables to which the function applies. Each variable name corresponds to one component of x. Variable names must be consistent across constraints.
name	Name for the objective function.

Details

Define linear objective function of the form

$$f(x) = a'x + d$$

.

Value

An object of class `linFun`.

See Also

The main function for solving constrained programming problems is `solvecop`.

Examples

```
### Linear programming with linear and quadratic constraints ###
### Example from animal breeding                              ###
### The mean breeding value BV is maximized whereas the      ###
### mean kinship in the offspring x'Qx+d is restricted       ###

data(phenotype)
data(myQ)

A   <- t(model.matrix(~Sex-1, data=phenotype))
A[,1:5]
val <- c(0.5, 0.5)
dir <- c("==","==")

mycop <- cop(f  = linfun(a=phenotype$BV, id=phenotype$Indiv, name="BV"),
             max= TRUE,
             lb = lbcon(0,  id=phenotype$Indiv),
             ub = ubcon(NA, id=phenotype$Indiv),
             lc = lincon(A=A, dir=dir, val=val, id=phenotype$Indiv),
             qc = quadcon(Q=myQ, d=0.001, val=0.035, name="Kinship", id=rownames(myQ)))

res <- solvecop(mycop, solver="cccp2", quiet=FALSE)

validate(mycop, res)

#            valid solver  status
#             TRUE  cccp2 optimal
#
#   Variable     Value      Bound    OK?
#   -------------------------------------
#   BV           0.7667 max        :
#   -------------------------------------
#   lower bounds all x  >=  lb     : TRUE
#   Sexfemale    0.5    ==  0.5    : TRUE
#   Sexmale      0.5    ==  0.5    : TRUE
#   Kinship      0.035  <=  0.035  : TRUE
#   -------------------------------------
```

---

myQ *Kinship Matrix*

Description

Kinship matrix of the cattle listed in data frame phenotype. This is an (almost) positive semidefinite matrix.

Usage

```
data(myQ)
```

Format

Matrix

`myQ1` *Kinship Matrix*

Description

Matrix needed to compute kinship at native alleles for the cattle listed in data frame phenotype. This is an (almost) positive semidefinite matrix.

Usage

```
data(myQ1)
```

Format

Matrix

`myQ2` *Kinship Matrix*

Description

Matrix needed to compute kinship at native alleles for the cattle listed in data frame phenotype. This is an (almost) positive semidefinite matrix.

Usage

```
data(myQ2)
```

Format

Matrix

`phenotype` *Phenotypes of Genotyped Cattle*

Description

Phenotypes of cattle.

Usage

```
data(phenotype)
```

Format

Data frame containing information on genotyped cattle. The columns contain the IDs of the individuals (`Indiv`), simulated breeding values (`BV`), simulated sexes (`Sex`), and genetic contributions from other breeds (`MC`).

`print.copValidation` *Print Validation of a Solution*

Description

Print the validation results for the solution of an optimization problem.

Usage

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

Arguments

`x`	The result of function validate.
`...`	Unused additional arguments.

Details

Print the validation results for the solution of an optimization problem.

Value

A list of class copValidation (invisible) with components:

summary	Data frame containing one row for each constraint with the value of the constraint in column Val, the bound for the constraint in column Bound, and column OK states if the constraint is fulfilled. The value of the objective function is shown in the first row. Additional rows contain the values of disabled constraints.
info	Data frame with component valid indicating if all constraints are fulfilled, component solver containing the name of the solver used for optimization, and component status describing the solution as reported by the solver.
var	Data frame with the values of the objective function and constraints at the optimum.
obj.fun	Named numeric value with value and name of the objective function at the optimum.

See Also

The main function for solving constrained programming problems is solvecop.

Examples

```
### Quadratic programming with linear constraints      ###
### Example from animal breeding                       ###
### where the mean kinship in the offspring is minized ###

data(phenotype)
data(myQ)

A   <- t(model.matrix(~Sex+BV-1, data=phenotype))
rownames(A) <- c("male.cont","female.cont", "Breeding.Value")
val <- c(0.5, 0.5, 0.40)
dir <- c("==","==",">=")

mycop <- cop(f  = quadfun(Q=myQ, d=0.001, name="Kinship", id=rownames(myQ)),
             lb = lbcon(0,  id=phenotype$Indiv),
             ub = ubcon(NA, id=phenotype$Indiv),
             lc = lincon(A=A, dir=dir, val=val, id=phenotype$Indiv))

res <- solvecop(mycop, solver="cccp", quiet=FALSE, trace=FALSE)

head(res$x)

Evaluation <- validate(mycop, res, quiet=TRUE)

print(Evaluation)

#             valid solver  status
#              TRUE   cccp optimal
```

```
#
#   Variable       Value      Bound    OK?
#   ---------------------------------------
#   Kinship        0.0322 min        :
#   ---------------------------------------
#   lower bounds   all x  >=  lb     : TRUE
#   male.cont      0.5    ==  0.5    : TRUE
#   female.cont    0.5    ==  0.5    : TRUE
#   Breeding.Value 0.4    >=  0.4    : TRUE
#   ---------------------------------------
```

---

quadcon *Quadratic Constraint*

Description

Define a quadratic constraint of the form

$$x'Qx + a'x + d \leq val$$

Usage

```
quadcon(Q, a=rep(0, nrow(Q)), d=0, dir="<=", val,
   id=1:nrow(Q), name="quadratic", use=TRUE)
```

Arguments

`Q`	Numeric symmetric matrix of the constraint coefficients.
`a`	Numeric vector.
`d`	Numeric value.
`dir`	Character string `"<="`.
`val`	Numeric threshold value, which is the upper bound for the quadratic function.
`id`	Vector defining the names of the variables to which the constraint applies. Each variable name corresponds to one component of `x`. Variable names must be consistent across constraints.
`name`	Name for the constraint.
`use`	Logical value indicating if the constraint should be included in the optimization problem. If `use=FALSE`, then constraint does not affect the result, but the value of the quadratic function will be reported by function validate.

Details

Define a quadratic inequality constraint of the form

$$x'Qx + a'x + d \leq val.$$

Vector `x` contains only the variables included in argument `id`.

Value

An object of class quadCon.

See Also

The main function for solving constrained programming problems is solvecop.

Examples

```
### Linear programming with linear and quadratic constraints ###
### Example from animal breeding                               ###
### The mean breeding value BV is maximized whereas the        ###
### mean kinship in the offspring x'Qx+d is restricted         ###

data(phenotype)
data(myQ)

A   <- t(model.matrix(~Sex-1, data=phenotype))
A[,1:5]
val <- c(0.5, 0.5)
dir <- c("==","==")

mycop <- cop(f  = linfun(a=phenotype$BV, id=phenotype$Indiv, name="BV"),
             max= TRUE,
             lb = lbcon(0,  id=phenotype$Indiv),
             ub = ubcon(NA, id=phenotype$Indiv),
             lc = lincon(A=A, dir=dir, val=val, id=phenotype$Indiv),
             qc = quadcon(Q=myQ, d=0.001, val=0.035, name="Kinship", id=rownames(myQ)))

res <- solvecop(mycop, solver="cccp2", quiet=FALSE)

validate(mycop, res)

#            valid solver  status
#             TRUE  cccp2 optimal
#
#   Variable     Value      Bound    OK?
#   -------------------------------------
#   BV           0.7667 max        :
#   -------------------------------------
#   lower bounds all x  >=  lb     : TRUE
#   Sexfemale    0.5    ==  0.5    : TRUE
#   Sexmale      0.5    ==  0.5    : TRUE
#   Kinship      0.035  <=  0.035  : TRUE
#   -------------------------------------
```

quadfun *Quadratic Objective Function*

Description

Define a quadratic objective function of the form

$$f(x) = x^T Q x + a^T x + d$$

Usage

```
quadfun(Q, a=rep(0, nrow(Q)), d=0, id=1:nrow(Q), name="quad.fun")
```

Arguments

Q	Numeric symmetric matrix of the constraint coefficients.
a	Numeric vector.
d	Numeric value.
id	Vector (if present), defining the names of the variables to which the function applies. Each variable name corresponds to one component of x. Variable names must be consistent across constraints.
name	Name for the objective function.

Details

Define a quadratic objective function of the form

$$f(x) = x^T Q x + a^T x + d$$

Value

An object of class quadFun.

See Also

The main function for solving constrained programming problems is solvecop.

Examples

```
### Quadratic programming with linear constraints       ###
### Example from animal breeding                        ###
### The mean kinship in the offspring x'Qx+d is minized ###
### and the mean breeding value is restricted.          ###

data(phenotype)
data(myQ)
```

```
A   <- t(model.matrix(~Sex+BV-1, data=phenotype))
A[,1:5]
val <- c(0.5, 0.5, 0.40)
dir <- c("==","==",">=")

mycop <- cop(f  = quadfun(Q=myQ, d=0.001, name="Kinship", id=rownames(myQ)),
             lb = lbcon(0,  id=phenotype$Indiv),
             ub = ubcon(NA, id=phenotype$Indiv),
             lc = lincon(A=A, dir=dir, val=val, id=phenotype$Indiv))

res <- solvecop(mycop, solver="cccp", quiet=FALSE)

validate(mycop, res)

#            valid solver  status
#             TRUE   cccp optimal
#
#   Variable     Value      Bound    OK?
#   ------------------------------------
#   Kinship      0.0322 min        :
#   ------------------------------------
#   lower bounds all x  >=  lb     : TRUE
#   Sexfemale    0.5    ==  0.5    : TRUE
#   Sexmale      0.5    ==  0.5    : TRUE
#   BV           0.4    >=  0.4    : TRUE
#   ------------------------------------
```

---

ratiocon *Rational Constraint*

Description

Define a rational constraint of the form

$$\frac{x^T Q_1 x + a_1^T x + d_1}{x^T Q_2 x + a_2^T x + d_2} \leq val$$

Usage

```
ratiocon(Q1, a1=rep(0, nrow(Q1)), d1=0, Q2, a2=rep(0, nrow(Q2)), d2=0, dir="<=", val,
   id=1:nrow(Q1), name="rational", use=TRUE)
```

Arguments

Q1	Numeric quadratic matrix.
a1	Numeric vector.
d1	Numeric value.
Q2	Numeric quadratic matrix.
a2	Numeric vector.

d2	Numeric value.
dir	Character string "<=".
val	Numeric threshold value, which is the upper bound for the rational function.
id	Vector defining the names of the variables to which the constraint applies. Each variable name corresponds to one component of x. Variable names must be consistent across constraints.
name	Name for the constraint.
use	Logical value indicating if the constraint should be included in the optimization problem. If use=FALSE, then the constraint does not affect the result, but the value of the rational function will be reported by function validate.

Details

Define a rational inequality constraint of the form

$$\frac{x^T Q_1 x + a_1^T x + d_1}{x^T Q_2 x + a_2^T x + d_2} \leq val.$$

Vector x contains only the variables included in argument id.

For rational constraints it is required that there is a linear constraint ensuring that sum(x) is a constant. Furthermore, the denominator must be non-negative.

Value

An object of class ratioCon.

See Also

The main function for solving constrained programming problems is solvecop.

Examples

```
### Constrained optimization with rational objective                ###
### function and linear and quadratic constraints                   ###
### Example from animal breeding                                    ###
### The mean kinship at native alleles in the offspring is minimized  ###
### The mean breeding value and the mean kinship are constrained      ###

data(phenotype)
data(myQ)
data(myQ1)
data(myQ2)

A <- t(model.matrix(~Sex+BV+MC-1, data=phenotype))
A[,1:5]
val <- c(0.5,  0.5,  0.4,  0.5 )
dir <- c("==", "==", ">=", "<=")
```

```
mycop <- cop(f  = quadfun(Q=myQ, d=0.001, name="Kinship", id=rownames(myQ)),
             lb = lbcon(0,  id=phenotype$Indiv),
             ub = ubcon(NA, id=phenotype$Indiv),
             lc = lincon(A=A, dir=dir, val=val, id=phenotype$Indiv),
             rc = ratiocon(Q1=myQ1, Q2=myQ2, d1=0.0004, d2=0.00025, val=0.040,
                           id=rownames(myQ1), name="nativeKinship")
             )

res <- solvecop(mycop, solver="slsqp", quiet=FALSE)

validate(mycop, res)

#            valid solver                status
#             TRUE  slsqp successful completion
#
#   Variable      Value      Bound    OK?
#   --------------------------------------
#   Kinship       0.0324 min        :
#   --------------------------------------
#   lower bounds  all x  >=  lb     : TRUE
#   Sexfemale     0.5    ==  0.5    : TRUE
#   Sexmale       0.5    ==  0.5    : TRUE
#   BV            0.4    >=  0.4    : TRUE
#   MC            0.4668 <=  0.5    : TRUE
#   nativeKinship 0.04   <=  0.04   : TRUE
#   --------------------------------------
```

`ratiofun` *Rational Objective Function*

Description

Define a rational objective function of the form

$$f(x)=\frac{x^TQ_1x+a_1x+d_1}{x^TQ_2x+a_2x+d_2}$$

Usage

```
ratiofun(Q1, a1=rep(0, nrow(Q1)), d1=0, Q2, a2=rep(0, nrow(Q2)), d2=0,
   id=1:nrow(Q1), name="ratio.fun")
```

Arguments

Q1	Numeric quadratic matrix.
a1	Numeric vector.
d1	Numeric value.
Q2	Numeric quadratic matrix.

a2	Numeric vector.
d2	Numeric value.
id	Vector defining the names of the variables to which the constraint applies. Each variable name corresponds to one component of x. Variable names must be consistent across constraints.
name	Name for the constraint.

Details

Define a rational ofjective function of the form

$$f(x) = \frac{x^T Q_1 x + a_1 x + d_1}{x^T Q_2 x + a_2 x + d_2}$$

Reasonable bounds for the variables should be provided because the function can have several local optima. Solvers 'slsqp' (the default) and 'alabama' are recommended.

Value

An object of class ratioFun.

See Also

The main function for solving constrained programming problems is solvecop.

Examples

```
### Constrained optimization with rational objective               ###
### function and linear and quadratic constraints                  ###
### Example from animal breeding                                   ###
### The mean kinship at native alleles in the offspring is minimized ###
### The mean breeding value and the mean kinship are constrained     ###

data(phenotype)
data(myQ)
data(myQ1)
data(myQ2)

Ax <- t(model.matrix(~Sex+BV+MC-1, data=phenotype))
Ax[,1:5]
val <- c(0.5,  0.5,  0.4,  0.5 )
dir <- c("==", "==", ">=", "<=")

mycop <- cop(f  = ratiofun(Q1=myQ1, Q2=myQ2, d1=0.0004, d2=0.00025,
                           id=rownames(myQ1), name="nativeKinship"),
             lb = lbcon(0,  id=phenotype$Indiv),
             ub = ubcon(NA, id=phenotype$Indiv),
             lc = lincon(A=Ax, dir=dir, val=val, id=phenotype$Indiv),
             qc = quadcon(Q=myQ, d=0.001, val=0.035,
                          name="Kinship", id=rownames(myQ)))
```

```
res <- solvecop(mycop, quiet=FALSE)

validate(mycop, res)

#            valid solver               status
#             TRUE  slsqp successful completion
#
#   Variable      Value      Bound    OK?
#   --------------------------------------
#   nativeKinship 0.0366 min        :
#   --------------------------------------
#   lower bounds  all x  >=  lb     : TRUE
#   Sexfemale     0.5    ==  0.5    : TRUE
#   Sexmale       0.5    ==  0.5    : TRUE
#   BV            0.4    >=  0.4    : TRUE
#   MC            0.4963 <=  0.5    : TRUE
#   Kinship       0.035  <=  0.035  : TRUE
#   --------------------------------------
```

solvecop — *Solve a Constrained Optimization Problem*

Description

Solve a constrained optimization problem with a linear, quadratic, or rational objective function, and linear, quadratic, rational, and boundary constraints.

Usage

```
solvecop(op, solver="default", make.definite=FALSE, X=NULL, quiet=FALSE, ...)
```

Arguments

- `op` — An optimization problem, usually created with function cop.
- `solver` — Character string with the name of the solver. Available solvers are `"alabama"`, `"cccp"`, `"cccp2"`, and `"slsqp"`. Solver `"csdp"` is temporarily disabled because the package Rcsdp has been removed from Cran. The default means that the solver is chosen automatically. The solvers are described in the Details section.
- `make.definite` — Logical variable indicating whether non-positive-semidefinite matrices should be approximated by positive-definite matrices. This is always done for solvers that are known not to convergue otherwise.
- `X` — Starting vector of parameter values (not needed). Any initial vector, even those violating linear inequality constraints, may be specified. Ignored by solvers `"cccp"` and `"csdp"`. For `"slsqp"` the lower and upper bounds must not be violated.
- `quiet` — Logical variable indicating whether output to console should be switched off.

...	Tuning parameters of the solver. The available parameters depend on the solver and will be printed when the function is used with quiet=FALSE. In section Details it is mentioned where descriptions of these parameters can be found.

Details

Solve a constrained optimization problem with a linear, quadratic, or rational objective function, and linear, quadratic, rational, and boundary constraints.

Solver

"alabama": The augmented lagrangian minimization algorithm auglag from package alabama is called. The method combines the objective function and a penalty for each constraint into a single function. This modified objective function is then passed to another optimization algorithm with no constraints. If the constraints are violated by the solution of this sub-problem, then the size of the penalties is increased and the process is repeated. The default methods for the uncontrained optimization in the inner loop is the quasi-Newton method called BFGS. Tuning parameters used for the outer loop are described in the details section of the help page of function auglag. Tuning parameters used for the inner loop are described in the details section of the help page of function optim.

"cccp" and "cccp2": Function cccp from package cccp for solving cone constrained convex programs is called. For solver "cccp", quadratic constraints are converted into second order cone constraints, which requires to approximate non-positive-semidefinite matrices by positive-definite matrices. For solver "cccp2", quadratic constraints are defined by functions. The implemented algorithms are partially ported from CVXOPT. Tuning parameters are those from function ctrl.

"slsqp": The sequential (least-squares) quadratic programming (SQP) algorithm slsqp for gradient-based optimization from package nloptr. The algorithm optimizes successive second-order (quadratic/least-squares) approximations of the objective function, with first-order (affine) approximations of the constraints. Available parameters are described in nl.opts

Value

A list with the following components:

x	Named numeric vector with parameters optimizing the objective function while satisfying constraints, if convergence is successful.
solver	Name of the solver used for optimization.
status	Message indicating type of convergence as reported by the solver.

Author(s)

Robin Wellmann

Examples

```
### Quadratic programming with linear constraints      ###
### Example from animal breeding                       ###
### where the mean kinship in the offspring is minized ###
```

```
data(phenotype)
data(myQ)

A   <- t(model.matrix(~Sex+BV-1, data=phenotype))
rownames(A) <- c("male.cont","female.cont", "Breeding.Value")
val <- c(0.5, 0.5, 0.40)
dir <- c("==","==",">=")

mycop <- cop(f  = quadfun(Q=myQ, d=0.001, name="Kinship", id=rownames(myQ)),
             lb = lbcon(0,  id=phenotype$Indiv),
             ub = ubcon(NA, id=phenotype$Indiv),
             lc = lincon(A=A, dir=dir, val=val, id=phenotype$Indiv))

res <- solvecop(mycop, solver="cccp", quiet=FALSE, trace=FALSE)

head(res$x)

hist(res$x,breaks=50,xlim=c(0,0.5))

Evaluation <- validate(mycop, res)

Evaluation$summary

Evaluation$info

Evaluation$obj.fun

Evaluation$var

Evaluation$var$Breeding.Value
```

---

ubcon *Upper Bounds*

Description

Define upper bounds for the variables of the form

$$x <= val.$$

Usage

```
ubcon(val=numeric(0), id=seq_along(val))
```

Arguments

`val`	Numeric vector with upper bounds for the variables. If `val` is a single value, then this value will be used for all variables in vector `id`.

id — Vector defining the names of the variables to which the constraint applies. Each variable name corresponds to one component of x. Variable names must be consistent across constraints.

Details

Define upper bounds for the variables of the form

$$x <= val.$$

Vector x contains only the variables included in argument id.

Value

An object of class ubCon.

See Also

The main function for solving constrained programming problems is solvecop.

Examples

```
### Linear programming with linear and quadratic constraints ###
### Example from animal breeding                              ###
### The mean breeding value BV is maximized whereas the      ###
### mean kinship in the offspring x'Qx+d is restricted       ###
### Lower and upper bounds for females are identical, so     ###
### their contributions are not optimized.                   ###
### Lower and upper bounds for some males are defined.       ###

data(phenotype)
data(myQ)

A   <- t(model.matrix(~Sex-1, data=phenotype))
A[,1:5]
val <- c(0.5,  0.5)
dir <- c("==","==")
Nf  <- sum(phenotype$Sex=="female")
id  <- phenotype$Indiv

lbval <- setNames(rep(0,  length(id)), id)
ubval <- setNames(rep(NA, length(id)), id)
lbval[phenotype$Sex=="female"] <- 1/(2*Nf)
ubval[phenotype$Sex=="female"] <- 1/(2*Nf)
lbval["276000102379430"] <- 0.02
ubval["276000121507437"] <- 0.03

mycop <- cop(f  = linfun(a=phenotype$BV, id=id, name="BV"),
             max= TRUE,
             lb = lbcon(lbval, id=id),
```

```
              ub = ubcon(ubval, id=id),
              lc = lincon(A=A, dir=dir, val=val, id=id),
              qc = quadcon(Q=myQ, d=0.001, val=0.045,
                           name="Kinship", id=rownames(myQ)))

res <- solvecop(mycop, solver="cccp2", quiet=FALSE)

Evaluation <- validate(mycop, res)

#             valid solver  status
#              TRUE  cccp2 optimal
#
#   Variable     Value      Bound    OK?
#   -------------------------------------
#   BV           0.5502 max        :
#   -------------------------------------
#   lower bounds all x  >=  lb     : TRUE
#   upper bounds all x  <=  ub     : TRUE
#   Sexfemale    0.5    ==  0.5    : TRUE
#   Sexmale      0.5    ==  0.5    : TRUE
#   Kinship      0.045  <=  0.045  : TRUE
#   -------------------------------------
```

---

validate *Validate a Solution*

Description

Validate a solution of an optimization problem.

Usage

```
validate(op, sol, quiet=FALSE, tol=0.0001)
```

Arguments

`op`	The constrained optimization problem defined with function cop.
`sol`	The solution of the optimization problem obtained with function solvecop.
`quiet`	Logical variable indicating whether output to console should be switched off.
`tol`	The tolerance. A constraint is considered fulfilled even if the value exceeds (falls below) the thresshold value by `tol`.

Details

Validate a solution of an optimization problem by checking if the constraints are fulfilled.

Values and bounds of the constraints are printed.

Value

A list of class copValidation with components:

summary: Data frame containing one row for each constraint with the value of the constraint in column Val, the bound for the constraint in column Bound, and column OK states if the constraint is fulfilled. The value of the objective function is shown in the first row. Additional rows contain the values of disabled constraints.

info: Data frame with component valid indicating if all constraints are fulfilled, component solver containing the name of the solver used for optimization, and component status describing the solution as reported by the solver.

var: Data frame with the values of the objective function and constraints at the optimum.

obj.fun: Named numeric value with value and name of the objective function at the optimum.

Author(s)

Robin Wellmann

See Also

The main function for solving constrained programming problems is solvecop.

Index