Package ‘fellov’

February 21, 2020

Type Package

Title Feasible Ellipse Overlap

Version 0.1

Author Anna Laksafoss

Maintainer Anna Laksafoss <vbs190@alumni.ku.dk>

Description A small package for determining if n-dimensional ellipses overlap.

Depends R (>= 3.6.0)

License GPL

Encoding UTF-8

LazyData true

Suggests testthat

RoxygenNote 7.0.2

NeedsCompilation no

Repository CRAN

Date/Publication 2020-02-21 14:10:05 UTC

R topics documented:

`feasible_overlap` *Find Smallest Feasible Ellipse Overlap*

Description

`feasible_overlap` will find the smallest radius such that the ellipses have a non-empty intersection.

Usage

```
feasible_overlap(ell, ...)
```

Arguments

`ell`	a list of at least two (non degenerate) ellipses; see `wrangle_ellipse`.
`...`	additional arguments to be passed to internal functions.

Details

Given a list of ellipses `ell` the function `feasible_overlap` will find the smallest radius such that the ellipses from `ell` overlap. This is done by solving the following quadratically constrained problem

$$\begin{array}{lll} \min_{(x,s)} & s & \\ \text{s.t.} & (x - c_i)^T P_i (x - c_i) - r_i \leq s & \text{for all i = 1, ..., d} \end{array}$$

To solve this convex problem the logarithmic barrier method is used.

Note that it is not necessary to specify ellipse radii in `ell`.

Value

`feasible_overlap` returns an object of `class` `"feasible_overlap"`. This object is a list with the following entries:

`radii`	the smallest ellipse radii resulting in a non-empty intersection.
`x`	The limiting common point.
`distance`	The ellipse specific distances.
`call`	The matched call.

See Also

`wrangle_ellipse` for detailed on ellipse parameterization.

Examples

```
## two dimensional ellipses
e1 <- list("c" = c(1,1), "P" = matrix(c(2,0,0,0.5), ncol = 2))
e2 <- list("c" = c(0,0), "S" = matrix(c(1, 0.2, 0.2, 2), ncol = 2), "r" = 1)
# note: it is not necessary to specify an ellipse radius "r"

feasible_overlap(list(e1, e2))


## regression example
# generate data
n <- 100
E <- rbinom(n, 1, 0.5)
X <- rnorm(n, E * 3, 1)
Y <- rnorm(n, 2 + 1.5 * X, 1)

# create confidence region ellipses
m0 <- lm(Y ~ X, data = data.frame(Y, X), subset = (E == 0))
m1 <- lm(Y ~ X, data = data.frame(Y, X), subset = (E == 1))
ConfRegion0 <- list(c = coefficients(m0), S = vcov(m0))
ConfRegion1 <- list(c = coefficients(m0), S = vcov(m0))

# find smallest radius
res <- feasible_overlap(list(ConfRegion0, ConfRegion1))
# this radius now corresponds to the chisq quantile at which
# the confidence regions intersect non-emptily.
# In other words the (1 - alpha)-confidence intervals intersect for alpha:
alpha <- pchisq(res$radii, 2)
```

`feasible_point` *Find Feasible Point in Ellipse Overlap*

Description

`feasible_point` will find a point in the interior of the intersection of two or more fully specified ellipses. If the intersections is empty `NA` is returned.

Usage

```
feasible_point(ell, ...)
```

Arguments

`ell`	a list of at least two (non degenerate) ellipses; see `wrangle_ellipse`.
`...`	additional arguments to be passed to internal functions.

Details

`feasible_point` will find a point in the interior of the intersection of two or more fully specified ellipses `ell`. If the intersections is empty `NA` is returned.

Value

`feasible_point` returns an object of `class` `"feasible_point"` with the following entries

`x`	An interior point.
`distance`	A data.frame with the ellipse specific distances.
`optim`	The final internal optimization value.
`call`	The matched call.

See Also

`wrangle_ellipse` for detailed on ellipse parameterization.

Examples

```
# two dimensional ellipses
e1 <- list("c" = c(1,2), "P" = matrix(c(2,0,0,1), ncol = 2), "r" = 3)
e2 <- list("c" = c(0,0), "S" = matrix(c(1, 0.2, 0.2, 2), ncol = 2), "r" = 1)

# find point in intersection
feasible_point(list(e1, e2))

# make new ellipse
e3 <- list("c" = c(2,2), "P" = matrix(c(1,0,0,1), ncol = 2), "r" = 0.5)

# now there is no overlap
feasible_point(list(e1, e2, e3))
```

`is_feasible_point` *Determine If A Point Is In Ellipse Overlap*

Description

`is_feasible_point` will determine if a given point is in the interior of the intersection of one or more fully specified ellipses.

Usage

```
is_feasible_point(point, ell)
```

Arguments

`point`	a numeric of length equal to the dimensions of the ellipses in `ell`.
`ell`	a list of at least one ellipse; see `wrangle_ellipse`.

Details

Given a point `is_feasible_point` will check if this point is in the intersection of the list of ellipses `ell`. Note that this function will not check if the intersection is non-empty.

Value

`is_feasible_point` returns an object of `class` `"is_feasible_point"`. This object is a list containing the following components:

`point`	the inputted `point`.
`fasible`	logical; is `TRUE` when the point `x` is in the interior of all ellipses.
`distance`	a data.frame with the distance from `x` to the center of each ellipse, the radius of each ellipse and a logical indicator, which is `TRUE` when `x` is an element in the ellipse.
`call`	the match call.

See Also

`wrangle_ellipse` for detailed on ellipse parameterization.

Examples

```
e1 <- list("c" = c(1,1), "P" = matrix(c(3,1,1,2), ncol = 2), "r" = 2)
e2 <- list("c" = c(0,2), "S" = matrix(c(4,1,1,1), ncol = 2), "r" = 3)

is_feasible_point(c(1.1,0.9), e1)
is_feasible_point(c(1,0), list(e1, e2))
```

`marginal_overlap` *Feasibility of all Marginal Ellipse Overlaps*

Description

Determin if the projections of ellipses onto each margin overlap.

Usage

```
marginal_overlap(ell, margins = "all")
```

Arguments

`ell`	a list of at least two (non degenerate) ellipses; see `wrangle_ellipse`.
`margins`	either "all" or a vector indicating the margins to project the ellipses onto and take intersections.

Details

The ellipses are projected onto the specified margins. For each margin the intersection of the projected ellipses is found. The Lower and Upper endpoints of the intersection interval is reported. If the intersection along a margin is empty then Lower and Upper is reported as NA.

Note that if the ellipses overlap when projected onto each margin this does not imply that the ellipses themselves intersect non-emptily. The example below is constructed to illustrate this.

Value

marginal_overlap returns an object of class "marginal_overlap" which contains a data.frame where the coloumns descibe the following

Margin	Intputted margins.
Overlap	Whether the ellipses overlap when projected onto corresponding margin.
Lower	Lower endpoint of intersection interval. NA if the intersection is empty.
Upper	Upper endpoint of intersection interval. NA if the intersection is empty.

See Also

wrangle_ellipse for detailed on ellipse parameterization.

Examples

```
## two dimensional ellipses
e1 <- list(c = c(0.1, 0), P = matrix(c(3, 0, 0, 1), ncol = 2), r = 1)
e2 <- list(c = c(1, 1), P = matrix(c(3, 1.2, 1.2, 1), ncol = 2), r = 0.8)
e3 <- list(c = c(2, 1.5), P = matrix(c(1, 0.6, 0.6, 1), ncol = 2), r = 0.4)
# Note: These three ellipses have been chosen so (some of) the marginal
#       projections intersect, but the actual ellipses do not intersect.

# Ellipses e1 and e2 overlap when projected onto margin 1 and 2 respectivly.
marginal_overlap(list(e1, e2))

# Adding ellipse e3:
# Then there is no overlap when projecting onto margin 1
marginal_overlap(list(e1, e2, e3), margins = c(1))


## regression example
n <- 100
E <- rbinom(n, 1, 0.5)
X <- rnorm(n, E * 3, 1)
Y <- rnorm(n, 2 + X, 1)
lm_E0 <- lm(Y ~ X, data = data.frame(Y, X), subset = (E == 0))
lm_E1 <- lm(Y ~ X, data = data.frame(Y, X), subset = (E == 1))

# create 95% confidence ellipses and check marginal overlap
q <- qchisq(0.95, 2) # df = 2, as there are two covariates (1, X)
ell0 <- list(c = coefficients(lm_E0), S = vcov(lm_E0), r = q)
ell1 <- list(c = coefficients(lm_E1), S = vcov(lm_E1), r = q)
marginal_overlap(list(ell0, ell1))
```

pairwise_overlap *Feasibility of all Pairwise Ellipse Overlaps*

Description

Determin if pairs of ellipses intersect non-emptily..

Usage

```
pairwise_overlap(ell, ...)
```

Arguments

`ell`	a list of at least two (non degenerate) ellipses; see `wrangle_ellipse`.
`...`	additional arguments to be passed to the low level funcitons.

Details

The `pairwise_overlap` functions goes through all pairs of ellipses from `ell` and checks if their intersection is non-empty.

Note that if all pairs of ellipses intersect this does not mean that the intersection of all the ellipses is non-empty. The example below is constructed to illustrate this.

Value

The `pairwise_overlap` function returns an object of `class` "`pairwise_overlap`" with the following components:

`intersection`	a data.frame where the two first columns specify the two ellipses intersected and the last coloumn indicate if they have a non-empty intersection.
`call`	the matched call.

See Also

`wrangle_ellipse` for detailed on ellipse parameterization.

Examples

```
## three different two dimensional ellipses
e1 <- list(c = c(0, 0.7), P = matrix(c(0.2, 0, 0, 3), ncol = 2), r = 0.5)
e2 <- list(c = c(0, 1), P = matrix(c(3, -1.5, -1.5, 1), ncol = 2), r = 1)
e3 <- list(c = c(1.5, 1), P = matrix(c(3, 1.2, 1.2, 1), ncol = 2), r = 1.2)
# Note: These ellipses have been chosen so all pairs intersect,
#       but the intersection of all three is empty.

# test pairwise overlaps
pairwise_overlap(list(e1, e2, e3))
```

```
## regression example
# generate data
n <- 100
E <- rbinom(n, 2, 0.5)
X <- rnorm(n, 3 * E, 1)
Y <- rnorm(n, 2 + 1.5 * E, 1)
m0 <- lm(Y ~ X, data = data.frame(Y,X), subset = (E == 0))
m1 <- lm(Y ~ X, data = data.frame(Y,X), subset = (E == 1))
m2 <- lm(Y ~ X, data = data.frame(Y,X), subset = (E == 2))

# create 95% confidence ellipses and check pairwise intersection
q <- qchisq(0.95, 2) # df = 2, as there are two covariates (1, X)
E0 <- list(c = coefficients(m0), S = vcov(m0), r = q)
E1 <- list(c = coefficients(m1), S = vcov(m1), r = q)
E2 <- list(c = coefficients(m2), S = vcov(m2), r = q)
pairwise_overlap(list("model 0" = E0, "model 1" = E1, "model 2" = E2))
```

wrangle_ellipse *Ellipse Wrangeler*

Description

wrangle_ellipse is used to wrangle one or more ellipses from one parametrization to another.

Usage

```
wrangle_ellipse(ell, out_params = c("c", "P", "r"))
```

Arguments

ell	a list of (non degenerate) ellipses to be wrangled. An ellipse is a named list and each entry corresponds to a parameter. To ensure all out_params can be calculated one of the parametrizations listed below in the description must be specified. Some out_params do not require a fully parametrized ellipse and so partially specified ellipses can be used.
out_params	a vector of names of the output parameters. A list of possible parameters is given below in the details.

Details

Takes ellipse parameters and and calculates the wanted out_params. A parameterization is a named list, where each named entry is a parameter. The following parameters are accepted both input and output:

- n : dimension of ellipse; an integer.
- c : center of the ellipse; a vector.
- P : precision matrix - inverse of S; a positive definit, symmetric matrix.

- S : deviation matrix - inverse of P; a positive definit, symmetric matrix.
- r : radius; a positive number.
- q : cross term -Pc; a vector.
- L : Cholesky decomposition of P
- e : eigen values of P; a vector of eigenvalues.
- U : eigen vectors of P; a matrix, where each column is an eigen vector.
- D : diagonal matrix with sqrt(e) as diagonal entries.

An ellipse E may be fully parameterized using the above parameters in the following ways:

$$E = \{x \in \mathsf{R}^p : (x - c)^T P(x - c) \leq r\}$$

$$E = \{x \in \mathsf{R}^p : (x - c)^T S^{-1}(x - c) \leq r\}$$

$$E = \{x \in \mathsf{R}^p : (x - c)^T LL^T(x - c) \leq r\}$$

$$E = \{x \in \mathsf{R}^p : (x - c)^T UD^2U^T(x - c) \leq r\}$$

$$E = \{x \in \mathsf{R}^p : ||L^T(x - c)||_2 \leq r\}$$

$$E = \{x \in \mathsf{R}^p : ||DU^T(x - c)||_2 \leq r\}$$

$$E = \{x \in \mathsf{R}^p : c + L^{-T}w,\ ||w|| \leq r\}$$

$$E = \{x \in \mathsf{R}^p : c + UD^{-1}w,\ ||w|| \leq r\}$$

To ensure that all of the above parameters can be calculated it is advised (but in some cases not needed) that the input ellipses are fully parameterized.

Value

A list of wrangled ellipses. The wrangled ellipses are now given by the out_params.

Examples

```
# two dimensional unite ball
e2d <- list(c = c(0,0), S = matrix(c(1,0,0,1), ncol = 2), r = 1)

# three dimensional ellipse
e3d <- list(c = c(3,2,1), P = matrix(c(3,1,2,1,5,0,2,0,2), ncol = 3))

f1 <- wrangle_ellipse(e2d) # (c,P,r) parameterization
f2 <- wrangle_ellipse(e2d, out_params = c("c", "e", "U", "r"))
f3 <- wrangle_ellipse(list("ellipse1" = e2d, "ellipse2" = e3d),
                      c("n", "c", "U", "D"))
```

Index