Package 'shotGroups'

July 23, 2025

Type Package

Title Analyze Shot Group Data

Version 0.8.2

Date 2022-09-17

Author Daniel Wollschlaeger

Maintainer Daniel Wollschlaeger <dwoll@kuci.org>

Depends R (>= 3.5.0)

Imports boot, coin, CompQuadForm (>= 1.4.2), graphics, grDevices, KernSmooth, robustbase, stats, tools, utils

Suggests knitr, markdown, energy, mvoutlier, shiny, jsonlite, interp, MBA, bs4Dash

LazyData yes

VignetteBuilder knitr

Description Analyzes shooting data with respect to group shape, precision, and accuracy. This includes graphical methods, descriptive statistics, and inference tests using standard, but also non-parametric and robust statistical methods. Implements distributions for radial error in bivariate normal variables. Works with files exported by 'OnTarget PC/TDS', 'Silver Mountain' e-target, 'ShotMarker' e-target, or 'Taran', as well as with custom data files in text format. Supports inference from range statistics such as extreme spread. Includes a set of web-based graphical user interfaces.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2022-09-17 19:06:04 UTC

Contents

`shotGroups-package` *Analyze shot group data*

Description

The `shotGroups` package provides functions to read in, plot, statistically describe, analyze, and compare shooting data with respect to group shape, precision, and accuracy. This includes graphical methods, descriptive statistics, and inference tests using standard, but also non-parametric and robust statistical methods. Works with files exported by 'OnTarget PC/TDS', 'Silver Mountain' e-target, 'ShotMarker' e-target, or 'Taran', as well as with custom data files in text format. Supports inference from range statistics such as extreme spread. Includes web-based graphical user interface.

Details

Package:	shotGroups
Type:	Package
Version:	0.8.2
Date:	2022-09-17
Depends:	R (>= 3.5.0)
Imports:	boot, coin, CompQuadForm (>= 1.4.2), graphics, grDevices, KernSmooth, robustbase, stats, tools, utils
Suggests:	knitr, energy, mvoutlier, shiny, jsonlite, interp, MBA, bs4Dash
License:	GPL (>= 2)

Use `help(package='shotGroups')` for a list of all functions and links to the detailed help pages with information on options, usage and output. For further explanantions and an example walk-through, see `vignette('shotGroups')`.

Author(s)

Daniel Wollschlaeger

Maintainer: Daniel Wollschlaeger <dwoll@kuci.org>

Examples

```
groupSpread(DFcciHV, dstTarget=100, conversion='yd2in', bootCI='none')
```

`analyzeGroup` *Analysis for a single group of bullet holes*

Description

Performs a comprehensive numerical and graphical analysis of a single group of bullet holes.

Usage

```
analyzeGroup(DF, xyTopLeft = TRUE, center = FALSE,
             dstTarget, conversion, bandW = 0.5,
             CEPtype = 'CorrNormal', bootCI = 'none')
```

Arguments

`DF`	a data frame containing (at least) either the variables `point.x`, `point.y` or `x`, `y` defining the bullet holes. Variables `distance` (distance to target), `aim.x`, `aim.y` (point of aim) are useful - if they are missing, a warning is given and a default assumed.
`xyTopLeft`	logical: is the origin of the absolute coordinate system in the top-left corner? See details.
`center`	logical: center groups to mean (0,0) first? If variable `series` does not exist, all shots are treated as belonging to the same group.
`dstTarget`	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable `distance` is already included in `DF`. See `getMOA`.
`conversion`	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables `dist.unit` and `point.unit` are already included in `DF`. Example `'m2cm'`. See `getMOA`.
`bandW`	for argument `bandwith` of `smoothScatter`.
`CEPtype`	string vector indicating which CEP estimate to report in `getCEP`.
`bootCI`	a character vector to select which bootstrap confidence interval type to report. Possible types are `'none'` (no bootstrap CI), `'norm'`, `'basic'`, `'perc'`, `'bca'`. See `boot.ci`.

Details

By default, OnTarget PC/TDS' 'Export Point Data' places the origin of the absolute coordinate system in the top-left corner. In OnTarget TDS, this setting can be changed by checking the box 'Tools -> Options -> Options tab -> Data Export -> Invert Y-Axis on Export'. In that case, use `xyTopLeft=FALSE`. If groups appear to be upside-down, `xyTopLeft` is the setting to change.

Robust estimates for the group center and the covariance matrix of (x,y)-coordinates are from `covMcd` using the MCD algorithm.

This function is a wrapper for groupShape, groupLocation, and groupSpread.
If the data is missing information about the point of aim, (0,0) is assumed. If distance to target is missing, 100 is assumed.
The number of replicates for the reported bootstrap confidence intervals is at least 1499. If the BCa interval is reported, it is at least the number of points.

In addition to the numerical results listed below, this function produces the following diagrams:

- a combined plot for multivariate outlier identification as produced by aq.plot
- a scatterplot of the (x,y)-coordinates together with group center, circle with average distance to center, 50%-confidence ellipse - the latter also based on a robust estimate for the covariance matrix
- a scatterplot of the (x,y)-coordinates together with the minimum bounding box, minimum enclosing circle, and maximum group spread
- a chi-square Q-Q-plot for eyeballing multivariate normality as produced by chisq.plot, including a reference line with intercept 0 and slope 1
- a heatmap of a 2D-kernel density estimate for the (x,y)-coordinates as produced by smoothScatter together with group center and error ellipse based on a robust estimate for the covariance matrix
- a Q-Q-plot of x-coordinates for eyeballing normality
- a Q-Q-plot of y-coordinates for eyeballing normality
- a histogram of x-coordinates including a fitted normal distribution as well as a non-parametric kernel density estimate
- a histogram of y-coordinates including a fitted normal distribution as well as a non-parametric kernel density estimate
- a histogram of distances to group center including a fitted Rayleigh distribution as well as a non-parametric kernel density estimate

Value

A list with the results from the numerical analyses and statistical tests.

corXY	correlation matrix of (x,y)-coordinates.
corXYrob	robust estimate of correlation matrix of (x,y)-coordinates.
Outliers	a vector of row indices for observations identified as outliers.
ShapiroX	Shapiro-Wilk-Test result for normality of x-coordinates.
ShapiroY	Shapiro-Wilk-Test result for normality of y-coordinates.
multNorm	E-statistic-Test result for multivariate normality of (x,y)-coordinates.
sdXY	standard deviations of x- and y-coordinates (in original measurement units, MOA, SMOA, milliradian).
sdXci	parametric and bootstrap confidence intervals for the standard deviation of x-coordinates (in original measurement units, MOA, SMOA, milliradian).
sdYci	parametric and bootstrap confidence intervals for the standard deviation of y-coordinates (in original measurement units, MOA, SMOA, milliradian).

sdXYrob	robust standard deviations of x- and y-coordinates (in original measurement units, MOA, SMOA, milliradian).
covXY	covariance matrix of (x,y)-coordinates.
covXYrob	robust estimate of covariance matrix of (x,y)-coordinates.
distToCtr	mean and median distance from points to their center as well as estimated Rayleigh parameters sigma (precision), radial standard deviation RSD, and mean radius MR (in original measurement units, MOA, SMOA, milliradian).
sigmaCI	95%-parametric and bootstrap confidence intervals for sigma (in original measurement units, MOA, SMOA, milliradian).
RSDci	95%-parametric and bootstrap confidence intervals for radial standard deviation RSD (in original measurement units, MOA, SMOA, milliradian).
MRci	95%-parametric and bootstrap confidence intervals for mean radius MR (in original measurement units, MOA, SMOA, milliradian).
maxPairDist	maximum pairwise distance between points (center-to-center, = maximum spread, in original measurement units, MOA, SMOA, milliradian).
groupRect	width and height of bounding box with diagonal and figure of merit FoM (average side length, in original measurement units, MOA, SMOA, milliradian).
groupRectMin	width and height of minimum-area bounding box with diagonal and figure of merit FoM (average side length, in original measurement units, MOA, SMOA, milliradian).
minCircleRad	radius for the minimum enclosing circle (in original measurement units, MOA, SMOA, milliradian).
confEll	length of semi-major and semi-minor axis of the 50%-confidence ellipse (in original measurement units, MOA, SMOA, milliradian).
confEllRob	length of semi-major and semi-minor axis of the 50%-confidence ellipse based on a robust estimate for the covariance matrix (in original measurement units, MOA, SMOA, milliradian).
confEllShape	aspect ratio and flattening of the 50%-confidence ellipse.
confEllShapeRob	aspect ratio and flattening of the 50%-confidence ellipse based on a robust estimate for the covariance matrix.
CEP	estimate(s) for the 50%-circular error probable (CEP, in original measurement units, MOA, SMOA, milliradian).
ctr	(x,y)-offset of group center relative to point of aim.
ctrXci	95%-parametric and bootstrap confidence intervals for center x-coordinate.
ctrYci	95%-parametric and bootstrap confidence intervals for center y-coordinate.
ctrRob	robust estimate of group center offset relative to point of aim (MCD algorithm).
distPOA	distance from group center to point of aim (in original measurement units, MOA, SMOA, milliradian).
distPOArob	distance from robust estimate of group center to point of aim (in original measurement units, MOA, SMOA, milliradian).
Hotelling	Hotelling's T^2-Test result from testing if group center equals point of aim.

See Also

groupShape, groupLocation, groupSpread, compareGroups, getDistToCtr, getMaxPairDist, getBoundingBox, getMinBBox, getMinCircle, getConfEll, getCEP, getRayParam, getMOA, smoothScatter, chisq.plot, aq.plot, pcout, qqnorm, hist, kernel, shapiro.test, mvnorm.etest, anova.mlm, boot, boot.ci, covMcd

Examples

```
data(DFinch)

# select combined data from only first 2 series
DF  <- subset(DFinch, series %in% 1:2)
res <- analyzeGroup(DF, conversion='yd2in', bootCI='none')
names(res)
res$multNorm
res$corXY
res$ctrRob
res$ctrXci
res$ctrYci
```

---

combineData *Combine list of data frames into one*

Description

Combines a list of data frames (the result from using readDataOT1, readDataOT2, or readDataMisc with combine=FALSE) into one big data frame.

Usage

```
combineData(DFs)
```

Arguments

DFs	a list of data frames with a shared set of variables.

Details

Assumes that the data frames in the list have a non-empty set of shared variables. Among them at least either point.x, point.y or x, y defining the bullet holes. To be useful for functions analyzeGroup or compareGroups, the data frames should also have variables group, distance, aim.x, aim.y defining point of aim. If group is missing, it is set to 1.

Value

A data frame with the shared set of variables. In addition, it also contains factors identifying the original file (`file`), and a factor identifying all groups from different data frames (`series`).

`...`	the shared set of variables from the the data frames in the list.
`group`	a factor that is the original `group` variable as defined by OnTarget PC/TDS.
`groupVerb`	a factor that codes group with more descriptive levels taken from the original project title, file name and ammunition (if available).
`file`	a factor that codes from which original file the data is.
`series`	a factor that codes each separate group across original files.
`seriesNum`	a factor that codes each separate group as a number that runs consecutively across original files.

See Also

`readDataMisc`, `readDataOT1`, `readDataOT2`, `analyzeGroup`, `compareGroups`

Examples

```
## combine list of data frames to one single data frame
data(DFlistCm)
DFcm <- combineData(DFlistCm)
str(DFcm)
head(DFcm)
```

---

compareGroups *Compare bullet hole groups*

Description

Numerically and graphically compare accuracy, precision, and distribution shape of up to 15 groups of bullet holes.

Usage

```
compareGroups(DF, plots = TRUE, xyTopLeft = TRUE, center = FALSE,
              ABalt = c('two.sided', 'less', 'greater'),
              Walt = c('two.sided', 'less', 'greater'),
              CEPtype = 'CorrNormal', CEPlevel = 0.5, CIlevel = 0.95,
              dstTarget, conversion)
```

Arguments

DF	a data frame containing (at least) these variables: series (a factor), and either point.x, point.y or x, y defining the bullet holes. Variables distance (distance to target), aim.x, aim.y (point of aim) are useful - if they are missing, a warning is given and a default assumed.
plots	logical: show diagrams?
xyTopLeft	logical: is the origin of the absolute coordinate system in the top-left corner? See details.
center	logical: center groups to mean (0,0) first to compare only with respect to precision?
ABalt	a character string indicating the hypothesis for the Ansari-Bradley-Test for equal variances. Only used when exactly 2 groups are compared.
Walt	a character string indicating the hypothesis for the Wilcoxon-Rank-Sum-Test for equality of average distance to group center (equivalent to the Mann-Whitney-U-Test). Only used when exactly 2 groups are compared.
CEPtype	string indicating which CEP estimate to report from getCEP.
CEPlevel	a numerical value giving the coverage of the confidence ellipse and CEP.
CIlevel	a numerical value giving the level for the confidence intervals (for standard deviations and Rayleigh sigma, MR).
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in DF. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables dist.unit and point.unit are already included in DF. Example 'm2cm'. See getMOA.

Details

By default, OnTarget PC/TDS' 'Export Point Data' places the origin of the absolute coordinate system in the top-left corner. In OnTarget TDS, this setting can be changed by checking the box 'Tools -> Options -> Options tab -> Data Export -> Invert Y-Axis on Export'. In that case, use xyTopLeft=FALSE. If groups appear to be upside-down, xyTopLeft is the setting to change.

OnTarget PC/TDS' Group variable identifies groups just within one file, whereas factor series is taken to number groups also across different original files. If your data was read with readDataOT1, readDataOT2 or readDataMisc, series is added automatically. For data from just one file, you can otherwise copy variable group to series in a data frame called shots with shots$series <- shots$group.
If the data is missing information about the point of aim, (0,0) is assumed. If distance to target is missing, 100 is assumed.

In addition to the numerical results listed below, this function produces the following diagrams:

- a scatterplot showing all groups as well as their respective center and confidence ellipse
- a scatterplot showing all groups as well as their respective (minimum) bounding box and maximum group spread
- a scatterplot showing all groups as well as their respective minimum enclosing circle and circle with average distance to center

- a boxplot for the distances to group center per group
- a stripchart showing the distances to group center per group together with the estimated Rayleigh mean radius and its confidence interval

If package `shiny` is installed, an interactive web app for this functionality can be run with `runGUI("analyze")`.

Value

A list with the results from numerical comparisons and statistical tests.

`ctr`	group center offset from the respective point of aim.
`distPOA`	distances from group centers to point of aim (in original measurement units, MOA, SMOA, milliradian).
`MANOVA`	MANOVA result from testing equality of group center offset from the respective point of aim (test statistic is Wilk's lambda).
`corXY`	group correlation matrices for the (x,y)-coordinates.
`sdXY`	list with group standard deviations of the x- and y-coordinates (in original measurement units, MOA, SMOA, milliradian).
`sdXYci`	list with group parametric (chi^2) confidence intervals for the standard deviations of x- and y coordinates (in original measurement units, MOA, SMOA, milliradian).
`meanDistToCtr`	average distances from points to their respective group center (in original measurement units, MOA, SMOA, milliradian).
`maxPairDist`	maximum pairwise distance between points for each group (center-to-center, = maximum spread, in original measurement units, MOA, SMOA, milliradian).
`bbFoM`	minimum-area bounding box figure of merit (average side length) for each group (in original measurement units, MOA, SMOA, milliradian).
`bbDiag`	minimum-area bounding box diagonal length for each group (in original measurement units, MOA, SMOA, milliradian).
`minCircleRad`	radius of the minimum enclosing circle for each group (in original measurement units, MOA, SMOA, milliradian).
`sigma`	estimated Rayleigh parameter sigma (precision) for each group (in original measurement units, MOA, SMOA, milliradian).
`MR`	estimated Rayleigh mean radius for each group (in original measurement units, MOA, SMOA, milliradian).
`sigmaMRci`	parametric (chi^2) confidence intervals for Rayleigh sigma and MR (in original measurement units, MOA, SMOA, milliradian).
`CEP`	Estimate for the circular error probable (CEP) in each group (in original measurement units, MOA, SMOA, milliradian).
`AnsariX`	Ansari-Bradley-Test result from testing equality of group variances for x-coordinates. When two groups are compared.
`AnsariY`	Ansari-Bradley-Test result from testing equality of group variances for y-coordinates. When two groups are compared.

`Wilcoxon`	Wilcoxon-Rank-Sum-Test result from testing equality of average point distances to their respective group center. When two groups are compared.
`FlignerX`	Fligner-Killeen-Test result from testing equality of group variances for x-coordinates. When more than two groups are compared.
`FlignerY`	Fligner-Killeen-Test result from testing equality of group variances for y-coordinates. When more than two groups are compared.
`Kruskal`	Kruskal-Wallis-Test result from testing equality of average point distances to their respective group center. When more than two groups are compared.

See Also

`analyzeGroup`, `getDistToCtr`, `getMaxPairDist`, `getMinBBox`, `getMinCircle`, `getCEP`, `getMOA`, `getRayParam`, `drawEllipse`, `anova.mlm`, `ansari_test`, `fligner_test`, `wilcox_test`, `kruskal_test`

Examples

```
cmp <- compareGroups(DF300BLKhl, dstTarget=100, conversion='yd2in')
names(cmp)
cmp$ctr
cmp$meanDistToCtr
cmp$CEP
cmp$Kruskal
```

DF300BLK *Combined bullet hole data*

Description

Example of a data frame from one file with one group of bullet holes.

Usage

```
data(DF300BLK)
```

Format

A data frame with 20 observations on the following 9 variables.

`group` a factor with level `1`.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`origin` a factor with level `1`. This factor codes from which original output file the data is.

`orgser` a factor with level `1.1`. This factor results from `droplevels(interaction(origin, group))`, and codes each separate group across original files. The order of the factor levels is alphabetical.

`series` a factor with level `1`. This factor codes each separate group as defined by `orgser`, but more conveniently as a number that runs consecutively across original files.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

One group of shooting a Noveske AR-15 rifle in 300BLK at 100yd with factory ammunition. The measurement unit for coordinates is inch, for distance yards.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.
Data courtesy of David Bookstaber, 2013. `http://ballistipedia.com/`

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DF300BLK)
str(DF300BLK)
```

DF300BLKhl *Combined bullet hole data*

Description

Example of a data frame from one file with one group of bullet holes.

Usage

```
data(DF300BLK)
```

Format

A data frame with 60 observations on the following 9 variables.

`group` a factor with level `1`.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`origin` a factor with level `1`. This factor codes from which original output file the data is.

`orgser` a factor with levels `1.1`, `1.2`, `1.3`. This factor results from `droplevels(interaction(origin, group))`, and codes each separate group across original files. The order of the factor levels is alphabetical.

`series` a factor with levels `1`, `2`, `3`. This factor codes each separate group as defined by `orgser`, but more conveniently as a number that runs consecutively across original files.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

Three groups of shooting a Noveske AR-15 rifle in 300BLK at 100yd with handloaded ammunition.
The measurement unit for coordinates is inch, for distance yards.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.
Data courtesy of David Bookstaber, 2014. `http://ballistipedia.com/`

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DF300BLK)
str(DF300BLK)
```

DFcciHV *Combined bullet hole data*

Description

Example of a data frame from one file with two groups of bullet holes.

Usage

```
data(DFcciHV)
```

Format

A data frame with 40 observations on the following 9 variables.

`group` a factor with levels `1` `2`.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`origin` a factor with levels `1`. This factor codes from which original output file the data is.

`orgser` a factor with levels `1.1 1.2`. This factor results from `droplevels(interaction(origin, group))`, and codes each separate group across original files. The order of the factor levels is alphabetical.

`series` a factor with levels `1 2`. This factor codes each separate group as defined by `orgser`, but more conveniently as a number that runs consecutively across original files.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

Two groups of shooting a PWS T3 rifle in .22LR at 100yd. The measurement unit for coordinates is inch, for distance yards.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.
Data courtesy of David Bookstaber, 2013. `http://ballistipedia.com/`

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DFcciHV)
str(DFcciHV)
```

DFcm *Combined bullet hole data*

Description

Example of a combined data frame from several files exported by OnTarget PC/TDS, each with several groups of bullet holes.

Usage

```
data(DFcm)
```

Format

A data frame with 487 observations on the following 13 variables.

`project.title` a character vector giving the OnTarget PC/TDS project title.

`group` a factor with levels `1 2 3`. This is the original `Group` variable as defined by OnTarget PC/TDS.

`ammunition` a character vector describing the ammo.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`center.x` a numerical vector of x-coordinates giving the group centers.

`center.y` a numerical vector of y-coordinates giving the group centers.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`origin` a factor with levels `1 2 3`. This factor codes from which original output file the data is.

`orgser` a factor with levels `1.1 2.1 3.1 1.2 2.2 3.2 1.3 2.3 3.3`. This factor results from `droplevels(interaction(origin, group))`, and codes each separate group across original files. The order of the factor levels is alphabetical.

`series` a factor with levels `1 2 3 4 5 6 7 8 9`. This factor codes each separate group as defined by `orgser`, but more conveniently as a number that runs consecutively across original files.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

`target` Character string `"BDS9"` indicating the target face. See `targets`.

Details

Several groups of shooting a 9x19mm pistol at 25m. The measurement unit for coordinates is cm, for distance meters.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DFcm)
str(DFcm)
```

DFdistr *Lookup table for distribution of range statistics and Rayleigh sigma*

Description

Lookup table for the distribution of range statistics and Rayleigh sigma from a Monte Carlo simulation of circular bivariate normal shot groups with 0 mean and variance 1 in both directions. Includes the first four moments and several quantiles of the distribution of extreme spread, figure of merit, bounding box diagonal, and Rayleigh sigma for each combination of number of shots per group and number of groups, repeated 10 million times.

Usage

```
data(DFdistr)
```

Format

A data frame with 590 observations on the following 77 variables.

n number of shots in each group. One of 2, 3, ..., 49, 50, 45, ..., 95, 100.

nGroups number of groups with individual simulated range statistics that were averaged over to yield the final value. One of 1, 2, ..., 9, 10.

nShots total number of shots, i.e., `n*nGroups`.

ES_M Extreme spread mean over all Monte Carlo simulations

ES_V Extreme spread variance over all Monte Carlo simulations

ES_SD Extreme spread standard deviation over all Monte Carlo simulations

ES_CV Extreme spread coefficient of variation over all Monte Carlo simulations

ESSQ_M Squard extreme spread mean over all Monte Carlo simulations

ESSQ_V Squared extreme spread variance over all Monte Carlo simulations

ES_SKEW Extreme spread skewness over all Monte Carlo simulations (smoothed)

ES_KURT Extreme spread kurtosis over all Monte Carlo simulations (smoothed)

ES_MED Extreme spread median (50% quantile) over all Monte Carlo simulations

ES_Q005 Extreme spread 0.5% quantile over all Monte Carlo simulations

ES_Q025 Extreme spread 2.5% quantile over all Monte Carlo simulations

ES_Q050 Extreme spread 5% quantile over all Monte Carlo simulations

ES_Q100 Extreme spread 10% quantile over all Monte Carlo simulations

ES_Q250 Extreme spread 25% quantile over all Monte Carlo simulations

ES_Q500 Extreme spread 50% quantile (median) over all Monte Carlo simulations

ES_Q750 Extreme spread 75% quantile over all Monte Carlo simulations

ES_Q900 Extreme spread 90% quantile over all Monte Carlo simulations

ES_Q950 Extreme spread 95% quantile over all Monte Carlo simulations

ES_Q975 Extreme spread 97.5% quantile over all Monte Carlo simulations

ES_Q995 Extreme spread 99.5% quantile over all Monte Carlo simulations

FoM_M Figure of merit mean over all Monte Carlo simulations

FoM_V Figure of merit variance over all Monte Carlo simulations

FoM_SD Figure of merit standard deviation over all Monte Carlo simulations

FoM_CV Figure of merit coefficient of variation over all Monte Carlo simulations

FoM_SKEW Figure of merit skewness over all Monte Carlo simulations (smoothed)

FoM_KURT Figure of merit kurtosis over all Monte Carlo simulations (smoothed)

FoM_MED Figure of merit median (50% quantile) over all Monte Carlo simulations

FoM_Q005 Figure of merit 0.5% quantile over all Monte Carlo simulations

FoM_Q025 Figure of merit 2.5% quantile over all Monte Carlo simulations
FoM_Q050 Figure of merit 0.25% quantile over all Monte Carlo simulations
FoM_Q100 Figure of merit 10% quantile over all Monte Carlo simulations
FoM_Q250 Figure of merit 25% quantile over all Monte Carlo simulations
FoM_Q500 Figure of merit 50% quantile (median) over all Monte Carlo simulations
FoM_Q750 Figure of merit 75% quantile over all Monte Carlo simulations
FoM_Q900 Figure of merit 90% quantile over all Monte Carlo simulations
FoM_Q950 Figure of merit 95% quantile over all Monte Carlo simulations
FoM_Q975 Figure of merit 97.5% quantile over all Monte Carlo simulations
FoM_Q995 Figure of merit 99.5% quantile over all Monte Carlo simulations
D_M Bounding box diagonal mean over all Monte Carlo simulations
D_V Bounding box diagonal variance over all Monte Carlo simulations
D_SD Bounding box diagonal standard deviation over all Monte Carlo simulations
D_CV Bounding box diagonal coefficient of variation over all Monte Carlo simulations
D_SKEW Bounding box diagonal skewness over all Monte Carlo simulations (smoothed)
D_KURT Bounding box diagonal kurtosis over all Monte Carlo simulations (smoothed)
D_MED Bounding box diagonal median (50% quantile) over all Monte Carlo simulations
D_Q005 Bounding box diagonal 0.5% quantile over all Monte Carlo simulations
D_Q025 Bounding box diagonal 2.5% quantile over all Monte Carlo simulations
D_Q050 Bounding box diagonal 5% quantile over all Monte Carlo simulations
D_Q100 Bounding box diagonal 10% quantile over all Monte Carlo simulations
D_Q250 Bounding box diagonal 25% quantile over all Monte Carlo simulations
D_Q500 Bounding box diagonal 50% quantile (median) over all Monte Carlo simulations
D_Q750 Bounding box diagonal 75% quantile over all Monte Carlo simulations
D_Q900 Bounding box diagonal 90% quantile over all Monte Carlo simulations
D_Q950 Bounding box diagonal 95% quantile over all Monte Carlo simulations
D_Q975 Bounding box diagonal 97.5% quantile over all Monte Carlo simulations
D_Q995 Bounding box diagonal 99.5% quantile over all Monte Carlo simulations
RS_M Rayleigh sigma mean over all Monte Carlo simulations
RS_V Rayleigh sigma variance over all Monte Carlo simulations
RS_SD Rayleigh sigma standard deviation over all Monte Carlo simulations
RS_CV Rayleigh sigma coefficient of variation over all Monte Carlo simulations
RS_SKEW Rayleigh sigma skewness over all Monte Carlo simulations (smoothed)
RS_KURT Rayleigh sigma kurtosis over all Monte Carlo simulations (smoothed)
RS_MED Rayleigh sigma median (50% quantile) over all Monte Carlo simulations
RS_Q005 Rayleigh sigma 0.5% quantile over all Monte Carlo simulations
RS_Q025 Rayleigh sigma 2.5% quantile over all Monte Carlo simulations

RS_Q050 Rayleigh sigma 5% quantile over all Monte Carlo simulations

RS_Q100 Rayleigh sigma 10% quantile over all Monte Carlo simulations

RS_Q250 Rayleigh sigma 25% quantile over all Monte Carlo simulations

RS_Q500 Rayleigh sigma 50% quantile (median) over all Monte Carlo simulations

RS_Q750 Rayleigh sigma 75% quantile over all Monte Carlo simulations

RS_Q900 Rayleigh sigma 90% quantile over all Monte Carlo simulations

RS_Q950 Rayleigh sigma 95% quantile over all Monte Carlo simulations

RS_Q975 Rayleigh sigma 97.5% quantile over all Monte Carlo simulations

RS_Q995 Rayleigh sigma 99.5% quantile over all Monte Carlo simulations

Details

The Monte Carlo distribution used 10 million repetitions in each scenario. One scenario was a combination of the `n` shots in each group, and the `nGroups` groups over which individual range statistics were averaged. Values for `n` were 2, 3, ..., 49, 50, 45, ..., 95, 100. Values for `nGroups` were 1, 2, ... 9, 10.

Skewness and kurtosis were smoothed using separate linear spline fits for each number of groups except for kurtosis of Rayleigh sigma which was fitted using the density of the gamma distribution.

Used in `range2sigma` to estimate Rayleigh parameter sigma from range statistics, and in `efficiency` to estimate the number of groups and total shots required to estimate the confidence interval (CI) for Rayleigh sigma with a given coverage probability (CI level) and width.

See the following source for an independent simulation, and for the rationale behind using it to estimate Rayleigh sigma:

http://ballistipedia.com/index.php?title=Range_Statistics

An older eqivalent simulation with less repetitions was done by Taylor and Grubbs (1975).

References

Taylor, M. S., & Grubbs, F. E. (1975). Approximate Probability Distributions for the Extreme Spread (BRL-MR-2438). Aberdeen Proving Ground, MD: U.S. Ballistic Research Laboratory.

See Also

`range2sigma`, `efficiency`, `getMaxPairDist`, `getBoundingBox`, `getRayParam`

Examples

```
data(DFdistr)
str(DFdistr)
```

DFinch *Combined bullet hole data*

Description

Example of a combined data frame from several files exported by OnTarget PC/TDS, each with several groups of bullet holes.

Usage

```
data(DFinch)
```

Format

A data frame with 487 observations on the following 13 variables.

`project.title` a character vector giving the OnTarget PC/TDS project title.

`group` a factor with levels `1 2 3`. This is the original `Group` variable as defined by OnTarget PC/TDS.

`ammunition` a character vector describing the ammo.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`center.x` a numerical vector of x-coordinates giving the group centers.

`center.y` a numerical vector of y-coordinates giving the group centers.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`origin` a factor with levels `1 2 3`. This factor codes from which original output file the data is.

`orgser` a factor with levels `1.1 2.1 3.1 1.2 2.2 3.2 1.3 2.3 3.3`. This factor results from `droplevels(interaction(origin, group))`, and codes each separate group across original files. The order of the factor levels is alphabetical.

`series` a factor with levels `1 2 3 4 5 6 7 8 9`. This factor codes each separate group as defined by `orgser`, but more conveniently as a number that runs consecutively across original files.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

`target` Character string `"BDS9"` indicating the target face. See `targets`

Details

Several groups of shooting a 9x19mm pistol at 27yd. The measurement unit for coordinates is inch, for distance yards.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.

See Also

combineData, analyzeGroup, compareGroups

Examples

```
data(DFinch)
str(DFinch)
```

DFlandy01 *Combined bullet hole data*

Description

Example of a data frame from one file with one group of bullet holes.

Usage

```
data(DFlandy01)
```

Format

A data frame with 530 observations on the following 15 variables.

`group` a numerical vector with group numbers 1 to 53.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`ammunition` a character vector giving ammunition and lot number.

`velocity` a numerical vector of chronograph readings in ft/s.

`control` undocumented.

`phase1` undocumented.

`phase2` undocumented.

`file` a character vector with the original file name.

`groupVerb` a character vector designating the group by combining the original file name and ammunition.

`series` a character vector that codes each separate group in an alternative way.

`seriesNum` like `series` but numeric.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

53 groups with 10 shots each of .22LR shot at Eley test center on Oct 2 2016 using a Stiller 2500X action at a distance of 50m. The measurement unit for coordinates is mm, for distance m.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.
Data courtesy of Larry Landercasper, 2017. Analyzed by Albert Highe.

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DFlandy01)
str(DFlandy01)
```

DFlandy02 *Combined bullet hole data*

Description

Example of a data frame from one file with one group of bullet holes.

Usage

```
data(DFlandy02)
```

Format

A data frame with 100 observations on the following 12 variables.

`group` a numerical vector with group numbers 1, 2.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`ammunition` a character vector giving ammunition and lot number.

`velocity` a numerical vector of chronograph readings in ft/s.

`file` a character vector with the original file name.

`groupVerb` a character vector designating the group by combining the original file name and ammunition.

`series` a character vector that codes each separate group in an alternative way.

`seriesNum` like `series` but numeric.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

2 groups with 50 shots each of .22LR shot using a Stiller 2500X action at a distance of 50yd. The measurement unit for coordinates is inch, for distance yards. Group 1 is from the best of 3 lots, group 2 from worst of 3 lots.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.
Data courtesy of Larry Landercasper, 2017.

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DFlandy02)
str(DFlandy02)
```

DFlandy03 *Combined bullet hole data*

Description

Example of a data frame from one file with one group of bullet holes.

Usage

```
data(DFlandy03)
```

Format

A data frame with 100 observations on the following 12 variables.

`group` a numerical vector with group numbers 1, 2.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`ammunition` a character vector giving ammunition and lot number.

`velocity` a numerical vector of chronograph readings in ft/s.

`file` a character vector with the original file name.

`groupVerb` a character vector designating the group by combining the original file name and ammunition.

`series` a character vector that codes each separate group in an alternative way.

`seriesNum` like `series` but numeric.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

4 groups with 25 shots each of .22LR shot at a distance of 50yd. The measurement unit for coordinates is inch, for distance yards.
Data courtesy of Larry Landercasper, 2017.

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DFlandy03)
str(DFlandy03)
```

`DFlandy04` *Combined bullet hole data*

Description

Example of a data frame from one file with one group of bullet holes.

Usage

```
data(DFlandy04)
```

Format

A data frame with 100 observations on the following 12 variables.

`group` a numerical vector with group numbers 1, 2.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`ammunition` a character vector giving ammunition and lot number.

`velocity` a numerical vector of chronograph readings in ft/s.

`file` a character vector with the original file name.

`groupVerb` a character vector designating the group by combining the original file name and ammunition.

`series` a character vector that codes each separate group in an alternative way.

`seriesNum` like `series` but numeric.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

6 groups with 25 shots each (groups 1-5) or 50 shots (group 6) of .22LR shot at a distance of 50yd. The measurement unit for coordinates is inch, for distance yards.
Groups 1-3 shot with a Stiller Copperhead action with Shilen Octagon Barrel. Group 4-5 shot with a Baity Falcon action with Shilen Ratchet Barrel. Group 6 shot with a Stiller 2500X action with Shilen Octagon Barrel.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.
Data courtesy of Larry Landercasper, 2017.

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DFlandy04)
str(DFlandy04)
```

DFlistCm *List containing several data frames with bullet hole data*

Description

Example list containing several data frames with bullet hole data as produced by `readDataOT1`, `readDataOT2`, or by `readDataMisc` with option `combine=FALSE`.

Usage

```
data(DFlistCm)
```

Details

Several groups of shooting a 9x19mm pistol at 25m. The measurement unit for coordinates is cm, for distance meters.
This list can be used as an argument for `combineData`.

See Also

`combineData`, `readDataMisc`, `readDataOT1`, `readDataOT2`

Examples

```
data(DFlistCm)
str(DFlistCm)

## combine list of data frames to one single data frame
DFcm <- combineData(DFlistCm)
str(DFcm)
```

DFsavage *Combined bullet hole data*

Description

Example of a combined data frame from several original files, each with one group.

Usage

```
data(DFsavage)
```

Format

A data frame with 180 observations on the following 10 variables.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`distance` a numerical vector giving the distance to the target.

`group` a factor with level `1`. This is the original `Group` variable as defined by OnTarget PC/TDS.

`bullet` a character vector describing the bullet type.

`origin` a factor with levels `1 ... 9`. This factor codes from which original output file the data is.

`orgser` a factor with levels `1.1 ... 9.1`. This factor results from `droplevels(interaction(Origin, Group))`, and codes each separate group across original files. The order of the factor levels is alphabetical.

`series` a factor with levels `1 ... 9`. This factor codes each separate group as defined by `orgser`, but more conveniently as a number that runs consecutively across original files.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

Several groups of shooting a Savage 12 FT/R rifle in .308 Win at distances from 100 to 300m. The measurement unit for coordinates is mm, for distance meters. Shots 1-5 in `series` 4, and shots 1-3 in `series` 7 moved the scope.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DFsavage)
str(DFsavage)
```

DFscar17 *Combined bullet hole data*

Description

Example of a data frame from one file with one group of bullet holes.

Usage

```
data(DFscar17)
```

Format

A data frame with 10 observations on the following 9 variables.

`group` a factor with level `1`.

`distance` a numerical vector giving the distance to the target.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`origin` a factor with level `1`. This factor codes from which original output file the data is.

`orgser` a factor with level `1.1`. This factor results from `droplevels(interaction(origin, group))`, and codes each separate group across original files. The order of the factor levels is alphabetical.

`series` a factor with levels `1`. This factor codes each separate group as defined by `orgser`, but more conveniently as a number that runs consecutively across original files.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

One group of shooting an FN SCAR 17 rifle in .308 Win at 100yd. The measurement unit for coordinates is inch, for distance yards.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.
Data courtesy of David Bookstaber, 2013. `http://ballistipedia.com/`

See Also

`combineData`, `analyzeGroup`, `compareGroups`

Examples

```
data(DFscar17)
str(DFscar17)
```

DFtalon *Combined bullet hole data*

Description

Example of a combined data frame from several original files, each file containing one group.

Usage

```
data(DFtalon)
```

Format

A data frame with 180 observations on the following 10 variables.

`point.x` a numerical vector of absolute x-coordinates of bullet holes.

`point.y` a numerical vector of absolute y-coordinates of bullet holes.

`aim.x` a numerical vector of x-coordinates giving the point of aim.

`aim.y` a numerical vector of y-coordinates giving the of point of aim.

`distance` a numerical vector giving the distance to the target.

`group` a factor with level `1`. This is the original `Group` variable as defined by OnTarget PC/TDS.

`bullet` a character vector describing the bullet type.

`origin` a factor with levels `1 ... 9`. This factor codes from which original output file the data is.

`orgser` a factor with levels `1.1 ... 9.1`. This factor results from `droplevels(interaction(Origin, Group))`, and codes each separate group across original files. The order of the factor levels is alphabetical.

`series` a factor with levels `1 ... 9`. This factor codes each separate group as defined by `orgser`, but more conveniently as a number that runs consecutively across original files.

`distance.unit` Measurement unit distance to target

`point.unit` Measurement unit (x,y)-coordinates

Details

Several groups of shooting a Talon SS air rifle at 10m. The measurement unit for coordinates is mm, for distance meters.
This data frame is like those returned by `readDataOT1`, `readDataOT2`, or `readDataMisc` with option `combine=TRUE`.

See Also

combineData, analyzeGroup, compareGroups

Examples

```
data(DFtalon)
str(DFtalon)
```

---

drawBox *Draw an axis-aligned box*

Description

Adds an axis-aligned box to an existing plot.

Usage

```
drawBox(x, fg = par('fg'), bg = NA,
        colCtr = NA, lty = par('lty'), lwd = par('lwd'),
        pch = par('pch'), cex = par('cex'))

## S3 method for class 'list'
drawBox(x, fg = par('fg'), bg = NA,
        colCtr = NA, lty = par('lty'), lwd = par('lwd'),
        pch = par('pch'), cex = par('cex'))

## Default S3 method:
drawBox(x, fg = par('fg'), bg = NA,
        colCtr = NA, lty = par('lty'), lwd = par('lwd'),
        pch = par('pch'), cex = par('cex'))
```

Arguments

x	either a list with component pts as returned by getBoundingBox, or a vector giving coordinates xleft, ybottom, xright, ytop.
fg	color of the box' rim.
bg	the box' fill color. Set to NA for a fully transparent box.
colCtr	color of the center point. Set to NA to omit.
lty	line type of the box.
lwd	line width of the box.
pch	symbol used for the center of the box.
cex	magnification factor for the symbol used for the center of the box.

Details

This function is mainly a wrapper for rect.

See Also

getBoundingBox, rect

Examples

```
xy  <- matrix(round(rnorm(20, 100, 15), 1), ncol=2)
(bb <- getBoundingBox(xy))

plot(xy, asp=1, pch=16)
drawBox(bb, fg='blue', colCtr='blue', pch=4, cex=2)
```

---

drawBox2 *Draw an oriented box*

Description

Adds an oriented box to an existing plot.

Usage

```
drawBox2(x, fg = par('fg'), bg = NA, colCtr = NA,
         lty = par('lty'), lwd = par('lwd'), pch = par('pch'),
         cex = par('cex'))

## S3 method for class 'list'
drawBox2(x, fg = par('fg'), bg = NA, colCtr = NA,
         lty = par('lty'), lwd = par('lwd'), pch = par('pch'),
         cex = par('cex'))

## Default S3 method:
drawBox2(x, fg = par('fg'), bg = NA, colCtr = NA,
         lty = par('lty'), lwd = par('lwd'), pch = par('pch'),
         cex = par('cex'))
```

Arguments

x	either a list with component pts as returned by getMinBBox, or a numerical (4 x 2)-matrix giving the (x,y)-coordinates of the ordered box vertices.
fg	color of the box' rim.
bg	the box' fill color. Set to NA for a fully transparent box.
colCtr	color of the center point. Set to NA to omit.
lty	line type of the box.
lwd	line width of the box.
pch	symbol used for the center of the box.
cex	magnification factor for the symbol used for the center of the box.

Details

This function is mainly a wrapper for polygon.

See Also

getMinBBox, polygon

Examples

```
xy  <- matrix(round(rnorm(20, 100, 15), 1), ncol=2)
(bb <- getMinBBox(xy))

plot(xy, xlim=range(c(xy[ , 1], bb$pts[ , 1])),
         ylim=range(c(xy[ , 2], bb$pts[ , 2])), asp=1, pch=16)
drawBox2(bb, fg='blue', colCtr='blue', pch=4, cex=2)
```

---

drawCircle *Draw a circle*

Description

Adds a circle to an existing plot.

Usage

```
drawCircle(x, radius, nv = 100, fg = par('fg'), bg = NA,
           colCtr = NA, lty = par('lty'), lwd = par('lwd'),
           pch = par('pch'), cex = par('cex'))

## S3 method for class 'list'
drawCircle(x, radius, nv = 100, fg = par('fg'), bg = NA,
           colCtr = NA, lty = par('lty'), lwd = par('lwd'),
           pch = par('pch'), cex = par('cex'))

## Default S3 method:
drawCircle(x, radius, nv = 100, fg = par('fg'), bg = NA,
           colCtr = NA, lty = par('lty'), lwd = par('lwd'),
           pch = par('pch'), cex = par('cex'))
```

Arguments

x	either a numerical vector giving the center's (x,y)-coordinates or a list with the components ctr and rad as returned by getMinCircle.
radius	a numerical vector giving the circle's radius.
nv	number of vertices in the approximating polygon.
fg	color of the circle's rim.

bg	the circle's fill color. Set to NA for a fully transparent circle.
colCtr	color of the center point. Set to NA to omit.
lty	line type of the circle.
lwd	line width of the circle.
pch	symbol used for the center of the circle.
cex	magnification factor for the symbol used for the center of the circle.

Details

This function is mainly a wrapper for polygon. To draw more than a few circles efficiently, use symbols instead.

See Also

polygon, symbols, getMinCircle

Examples

```
c1 <- c(1, 2)                # circle center
c2 <- c(2, 3)                # another circle center
r1 <- 2                      # circle radius
r2 <- 0.5                    # another circle radius

# determine axis limits so that circles will be visible
xLims <- c1[1] + c(-r1, r1)
yLims <- c1[2] + c(-r1, r1)

plot(c1[1], c1[2], type='n', asp=1, xlim=xLims, ylim=yLims)
drawCircle(c1, r1, fg='blue', colCtr='blue', pch=19)
drawCircle(c2, r2, fg='red', bg='red', colCtr='black', pch=4)
```

---

drawEllipse *Draw an ellipse*

Description

Adds an ellipse to an existing plot.

Usage

```
drawEllipse(x, shape, radius, nv = 100, axes = FALSE,
            fg = par('fg'), bg = NA, colCtr = NA, lty = par('lty'),
            lwd = par('lwd'), pch = par('pch'), cex = par('cex'))

## S3 method for class 'list'
drawEllipse(x, shape, radius, nv = 100, axes = FALSE,
            fg = par('fg'), bg = NA, colCtr = NA, lty = par('lty'),
```

```
            lwd = par('lwd'), pch = par('pch'), cex = par('cex'))

## Default S3 method:
drawEllipse(x, shape, radius, nv = 100, axes = FALSE,
            fg = par('fg'), bg = NA, colCtr = NA, lty = par('lty'),
            lwd = par('lwd'), pch = par('pch'), cex = par('cex'))
```

Arguments

x	either a numerical 2-vector giving the (x,y)-coordinates of the center or a list with the components ctr, cov and magFac as returned by getConfEll or getMinEllipse.
shape	a numerical symmetric (2 x 2)-matrix whose eigen-structure determines the ellipse's shape.
radius	a numerical value giving the ellipse's magnification factor.
nv	number of vertices in the approximating polygon.
axes	logical: should ellipse axes be drawn?
fg	color of the ellipse's rim.
bg	the ellipse's fill color. Set to NA for a fully transparent ellipse.
colCtr	color of the center point. Set to NA to omit.
lty	line type of the ellipse.
lwd	line width of the ellipse.
pch	symbol used for the center of the ellipse.
cex	magnification factor for the symbol used for the center of the ellipse.

Details

This function is mainly a wrapper for polygon.

See Also

polygon, getConfEll, getMinEllipse

Examples

```
## error ellipse for a set of points
xy <- matrix(round(rnorm(100, 0, 8), 2), ncol=2)
ce <- getConfEll(xy)
plot(xy, pch=16, asp=1)
drawEllipse(ce, radius=1, axes=TRUE, fg='blue', colCtr='blue',
            lwd=2, pch=4, cex=2)
```

drawGroup *Draw a group of bullet holes with additional measures*

Description

Draws a group with scaled bullet holes on a target background. Spread measures can be selected individually.

Usage

```
drawGroup(xy, center = FALSE, xyTopLeft = TRUE,
          bb = FALSE, bbMin = FALSE, bbDiag = FALSE,
          minCirc = FALSE, minEll = FALSE, maxSpread = FALSE,
          meanDist = FALSE, confEll = FALSE,
          CEP = FALSE, ringID = FALSE, valueID = TRUE, doRob = FALSE,
          level = 0.95, scaled = TRUE, caliber = 9, dstTarget, conversion,
          unit = 'unit', alpha = 0.5, target)

## S3 method for class 'data.frame'
drawGroup(xy, center = FALSE, xyTopLeft = TRUE,
          bb = FALSE, bbMin = FALSE, bbDiag = FALSE,
          minCirc = FALSE, minEll = FALSE,
          maxSpread = FALSE, meanDist = FALSE, confEll = FALSE,
          CEP = FALSE, ringID = FALSE, valueID = TRUE, doRob = FALSE,
          level = 0.95, scaled = TRUE, caliber = 9, dstTarget, conversion,
          unit = 'unit', alpha = 0.5, target)

## Default S3 method:
drawGroup(xy, center = FALSE, xyTopLeft = TRUE,
          bb = FALSE, bbMin = FALSE, bbDiag = FALSE,
          minCirc = FALSE, minEll = FALSE,
          maxSpread = FALSE, meanDist = FALSE, confEll = FALSE,
          CEP = FALSE, ringID = FALSE, valueID = TRUE, doRob = FALSE,
          level = 0.95, scaled = TRUE, caliber = 9, dstTarget, conversion,
          unit = 'unit', alpha = 0.5, target)
```

Arguments

xy	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n points (1 row of coordinates per point), or a data frame with either the variables `x`, `y` or `point.x`, `point.y` as well as `aim.x`, `aim.y` giving the point of aim. If missing, point of aim is assumed to be in (0,0).
center	logical: center groups to mean (0,0) first? If variable `series` does not exist, all shots are treated as belonging to the same group. Only available in method `drawGroup.data.frame()`.

xyTopLeft	logical: is the origin of the absolute coordinate system in the top-left corner? This is the default for data exported by OnTarget PC/TDS. If an (n x 2)-matrix is supplied for xy, point of aim is assumed to be in (0,0).
bb	logical: draw bounding box?
bbMin	logical: draw minimum-area bounding box?
bbDiag	logical: draw bounding box diagonal?
minCirc	logical: draw minimum enclosing circle?
minEll	logical: draw minimum enclosing ellipse?
maxSpread	logical: draw maximum spread?
meanDist	logical: draw circle with mean distance to group center?
confEll	logical: draw confidence ellipse with coverage level?
CEP	draw estimate of CEP circle with coverage level? Either logical or a string defining the CEP type. See getCEP.
ringID	logical: identify and display the ring count for each shot?
valueID	logical: display numerical values of calculated measures in the diagram?
doRob	logical: use robust estimation of group center and confidence ellipse?
scaled	logical: draw bullet holes to scale?
caliber	a numerical value indicating the bullet diameter in mm.
level	a numerical vector giving the coverages of the confidence ellipses and CEPs.
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in xy. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables dist.unit and point.unit are already included in xy. Example 'm2cm'. See getMOA.
unit	Measurement unit for the diagram. Defalt 'unit' indicates that the measurement unit given in conversion should be used. Possible values are 'unit', 'm', 'cm', 'mm', 'yd', 'ft', 'in', 'deg', 'MOA', 'SMOA', 'rad', 'mrad', 'mil'.
alpha	a numerical value in [0,1] which controls the alpha blending for simulated transparency used to draw the bullet holes
target	a character string like 'ISSF_100m' indicating the target type to be drawn in the scatterplot. See targets.

Value

Invisibly returns a list with the following components, all converted to unit (if they were requested):

xy	(x,y)-coordinates converted to unit.
ctr	(x,y)-offset of group center relative to point of aim in unit (robust with doRob=TRUE).
bb	bounding box as returned by getBoundingBox.
bbMin	minimum-area bounding box as returned by getMinBBox.
bbDiag	length of diagonal of bounding box.

bbMinDiag	length of diagonal of minimum-area bounding box.
minCirc	minimum enclosing circle as returned by getMinCircle.
minEll	minimum enclosing ellipse as returned by getMinEllipse.
maxPairDist	maximum pairwise distance between points (center-to-center, = maximum spread).
meanDist	mean distance to group center.
confEll	confidence ellipse with coverage level as returned by getConfEll (robust with doRob=TRUE).
CEP	Rayleigh estimate for the circular error probable CEP with coverage level.
target	Definition of the selected target in original and converted measurement units.
ringCount	Simulated and maximum ring count as returned by simRingCount.

See Also

getBoundingBox, getMinBBox, getMinCircle, getMinEllipse, getMaxPairDist, getDistToCtr, getConfEll, drawBox, drawBox2, drawCircle, drawEllipse, targets, drawTarget, simRingCount, covMcd

Examples

```
# draw group in MOA
dg <- drawGroup(DFcciHV, xyTopLeft=TRUE, bb=TRUE, minCirc=TRUE,
                confEll=TRUE, maxSpread=TRUE, caliber=5.56, unit='MOA',
                dstTarget=100, conversion='yd2in', target='BDS9')

# mininum enclosing circle in MOA
dg$minCirc

# show Grubbs-Patnaik CEP estimator for mulitple levels
drawGroup(DF300BLKhl, CEP="GrubbsPatnaik", level=c(0.5, 0.9, 0.95),
          dstTarget=100, conversion="yd2in", caliber=7.62)
```

---

drawTarget *Draw a target pattern*

Description

Draws a target pattern - either from the name of a pre-specified target type or from a supplied list defining the target.

Usage

```
drawTarget(x, unit, dstTarget, conversion,
           add = FALSE, cex = par('cex'))
```

Arguments

x	either a character value with the name of a target in targets or a list with a target definition containing the same components as those in targets (see below).
unit	the measurement unit that should be used in the plot. Possible values are 'cm', 'mm', 'm', 'in', 'ft', 'yd', 'deg', 'MOA', 'SMOA', 'rad', 'mrad', 'mil'.
dstTarget	a numerical value with the distance to the target - used in MOA calculation. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates - used in MOA calculation. Example 'm2cm'. See getMOA.
add	logical: add to existing plot or create new plot?
cex	numerical value specifying the magnification factor for plotting the ring numbers.

Value

Invisibly returns a list with (at least) the following components defining the target:

name	target name.
unitTarget	measurement unit for ring diameters and radii.
nRings	number of rings.
maxCount	highest ring count for scoring.
ringD10	diameter of ring number 10 (highest-valued ring).
ringD10i	diameter of sub-division of ring number 10 (Innenzehn). If target has no sub-division, equal to ringD10.
ringW	width of the remaining rings number 9, 8, 7, ...
cols	nRings+1 colors of the rings - right half of the target, starting with the sub-division of ring number 10 and going outwards.
colsTxt	nRings-1 colors of the ring numbers, starting with ring number 9 and going outwards.
ringR	nRings+1 ring radii, including sub-division of ring number 10.
unitConv	measurement unit for ringD10u, ringD10iu, ringWu, ringRu as defined by unit.
ringD10u	diameter of ring number 10 converted to unit.
ringD10iu	diameter of sub-division of ring number 10 (Innenzehn) converted to unit.
ringWu	width of the remaining rings number 9, 8, 7, ... converted to unit.
ringRu	nRings+1 ring radii, including sub-division of ring number 10, converted to unit.

See Also

targets, drawGroup

Examples

```
# draw ISSF 300m target in inch
trgt <- drawTarget('ISSF_300m', unit='in')

# target definition
trgt
```

`efficiency` *Estimate number of required groups for given CI level and width*

Description

Estimates the approximate number of required groups for a given number of shots per group, confidence interval (CI) level and CI width - when using the Rayleigh sigma estimator based on (x,y)-coordinates, or a range statistic such as extreme spread, figure of merit, or the bounding box diagonal. The function may also be used to obtain the estimated CI width when the number of shots per group and the number of groups is given. This functions assumes a circular bivariate normal shot distribution with 0 mean.

Usage

```
efficiency(n, nGroups, CIlevel=0.95, CIwidth,
           stat=c("Rayleigh", "ES", "FoM", "D"))
```

Arguments

`n`	a vector of integers between 2 and 100. Number of shots in each group.
`nGroups`	integer between 1 and 10. Number of groups over which individually-measured statistics will be averaged - when given, `CIwidth` must be missing, and the estimated CI width that can be achieved with the given `n`, `nGroups` and chosen statistic is returned.
`CIlevel`	confidence level - coverage probability of the CI.
`CIwidth`	CI width as a fraction of the mean of the chosen statistic for given `n` and `nGroups`. In other sources (see details) `CIwidth/2` is also called `E` - the width as a fraction of the mean on either side. When given, `nGroups` must be missing, and the estimated required number of groups to achieve the desired CI width for the CI coverage probability `CIlevel` is returned.
`stat`	a character vector with elements `"Rayleigh"` (Rayleigh sigma), `"ES"` (extreme spread), `"FoM"` (figure of merit), or `"D"` (bounding box diagonal) indicating which statistic would be measured.

Details

Based on the lookup table DFdistr with results form a Monte Carlo simulation. If the value of n is not among those simulated (but is less than 100), a spline interpolation between the neighboring simulated values of the statistic's coefficient of variation is used.

The number of required groups is approximate as the calculation assumes a normal distribution for the mean statistic. Details for the calculation can be found under

http://ballistipedia.com/index.php?title=Range_Statistics

http://www.geoffrey-kolbe.com/articles/rimfire_accuracy/group_statistics.htm

http://ballistipedia.com/images/3/32/Sitton_1990.pdf

If package shiny is installed, an interactive web app for this functionality can be run with runGUI("range").

Value

When CIwidth is given, a data frame with the estimated number of required groups and total shots to achieve the given CIlevel.

n	Number of shots per group.
nGroupsReq	Number of required groups - as calculated (fractional number).
nGroupsReqCeil	Number of required groups - rounded upwards to a whole number.
nShotsReq	Total number of required shots - using nGroupsReq as calculated (fractional number).
nShotsReqCeil	Total number of required shots - using nGroupsReq rounded upwards to a whole number.
CIlevel	The CI level
CIwidth	The CI width

When nGroups is given, a data frame with the estimated CI width required to achieve the desired CIlevel.

n	Number of shots per group.
nGroups	Number of groups that will be averaged over.
nShots	The total number of shots
CIlevel	The CI level
CIwidth	Required CI width as a fraction of the mean statistic.

See Also

DFdistr, range2sigma, getRayParam, getMaxPairDist, getBoundingBox

Examples

```
# get required number of shots to achieve 90% CI with
# a CI width of 20% of the mean (10% on either side)
# using 10 shots per group and measuring extreme spread
efficiency(n=10, CIlevel=0.9, CIwidth=0.2, stat="ES")

# as above, but using Rayleigh sigma
efficiency(n=10, CIlevel=0.9, CIwidth=0.2, stat="Rayleigh")

# check that the result for ES is about right
# -> 5% quantile with 10 groups is about 10% below the mean
# -> 95% quantile with 10 groups is about 10% above the mean
with(subset(DFdistr, (n == 10L) & (nGroups == 10L)),
     c(ES_Q050/ES_M, ES_Q950/ES_M))

# get achievable 90% CI width with 10 groups of 5 shots each
# using extreme spread
efficiency(n=5, nGroups=10, CIlevel=0.9, stat="ES")
```

---

fromMOA *Conversion from angular diameter to absolute size*

Description

Converts angular diameter (degree, radian, minute of angle MOA = arcminute, Shooter's MOA SMOA, milliradian mrad, NATO mil) to object size.

Usage

```
fromMOA(x, dst, conversion,
        type = c('deg', 'rad', 'MOA', 'SMOA', 'mrad', 'mil'))
```

Arguments

x	a numerical vector of angles.
dst	a numerical vector of viewing distances.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates. Either a character vector indicating the conversion such as 'm2cm' for meter to centimeter, 'yd2in' for yards to inches, or 'ft2cm' for feet to cm. Imperial (yd, ft, in) and metric units (m, cm, mm) can be freely mixed. Alternatively, a numerical vector giving the multiplication factor for conversion: 100 for m to cm, 36 for yd to in, and 12 for ft to in.
type	type of angular diameter: 'deg' for degree, 'rad' for radian, 'MOA' for minute of angle, 'SMOA' for Shooter's MOA, 'mrad' for milliradian, 'mil' for NATO mil. See details.

Details

1 MOA (minute of angle, arcmin) = 1/60 degree. Shooter's MOA = SMOA = Inches Per Hundred Yards IPHY. 1 inch at 100 yards = 1 SMOA. 1 milliradian = 1/1000 radian. 1 mil = 2*pi/6400 radian (NATO definition: the circle circumference is divided into 6400 mils). Details are given in the vignette, see `vignette('shotGroups')` .

If package `shiny` is installed, an interactive web app for this functionality can be run with `runGUI("angular")`.

Value

A numerical vector with the object sizes. The measurement unit is determined by `conversion`.

See Also

`getMOA`, `getDistance`

Examples

```
size <- seq(1, 20, by=5)   # inch
dst  <- 100                # yard
fromMOA(size, dst=dst, conversion='yd2in', type='MOA')

# this should return objSize
MOA <- getMOA(size, dst=dst, conversion='yd2in', type='MOA')
fromMOA(MOA, dst=dst, conversion='yd2in', type='MOA')

# SMOA
fromMOA(c(1, 2, 5), dst=100, conversion='yd2in', type='SMOA')

# milliradian
fromMOA(c(1, 2, 5), dst=100, conversion='m2mm', type='mrad')
```

---

getBoundingBox *Bounding box for a set of 2D-points*

Description

Calculates the vertices of the (axis-parallel) bounding box given a set of 2D-coordinates.

Usage

```
getBoundingBox(xy)

## S3 method for class 'data.frame'
getBoundingBox(xy)

## Default S3 method:
getBoundingBox(xy)
```

Arguments

xy	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n points (1 row of coordinates per point), or a data frame with either the variables x, y or point.x, point.y.

Details

No coordinate transforms are done (unlike in groupLocation, groupShape, groupSpread), i.e., origin is not assumed to be in top-left corner, and points are not taken relative to point of aim.

Value

A list with the following information about the bounding box:

pts	a numerical 4-vector giving the coordinates xleft, ybottom, xright, ytop.
width	width of the box.
height	height of the box.
FoM	figure of merit, i.e., the average side length of the box: (width + height) / 2.
diag	length of box diagonal.

See Also

drawBox, getMinBBox, getMinCircle

Examples

```
# coordinates given by a suitable data frame
bb <- getBoundingBox(DFsavage)

# draw points and bounding box
plot(point.y ~ point.x, data=DFsavage, asp=1, pch=16)
drawBox(bb, fg='blue', colCtr='blue', pch=4, cex=2)

bb$FoM                                  # figure of merit

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(20, 100, 15), 1), ncol=2)
getBoundingBox(xy)

## End(Not run)
```

getCEP *Circular Error Probable (CEP) and Spherical Error Probable (SEP)*

Description

Estimates the Circular Error Probable (CEP) or the Spherical Error Probable (SEP). CEP/SEP is defined as the radius of the circle/sphere around the point of aim such that it contains a certain fraction of points of impact, e.g., 50% or 90%.

Usage

```
getCEP(xy, CEPlevel=0.5, dstTarget, conversion,
       center = FALSE, accuracy = FALSE, type = 'CorrNormal', doRob = FALSE)

## S3 method for class 'data.frame'
getCEP(xy, CEPlevel=0.5, dstTarget, conversion,
       center = FALSE, accuracy = FALSE, type = 'CorrNormal', doRob = FALSE)

## Default S3 method:
getCEP(xy, CEPlevel=0.5, dstTarget, conversion,
       center = FALSE, accuracy = FALSE, type = 'CorrNormal', doRob = FALSE)
```

Arguments

xy	either a numerical (n x p)-matrix with the coordinates of n points (1 row of p coordinates per point), or a data frame with either the variables x, y or point.x, point.y (optionally z or point.z).
CEPlevel	a numerical vector with the coverage values for the CEP/SEP.
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in xy. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables dist.unit and point.unit are already included in xy. Example 'm2cm'. See getMOA.
center	logical: center groups to mean (0,0) first? If variable series does not exist, all shots are treated as belonging to the same group. Only available in method getCEP.data.frame().
accuracy	logical: take systematic location bias into account? See details.
type	string vector indicating which CEP/SEP estimate to report. Possible values are 'CorrNormal', 'GrubbsPearson', 'GrubbsPatnaik', 'GrubbsLiu', 'Rayleigh', 'Krempasky', 'Ignani', 'RMSE', 'Ethridge', 'RAND', 'Valstar'. See details.
doRob	logical: use robust estimation of center and covariance matrix as basis for estimators?

Details

For accuracy=FALSE (default), the reported CEP/SEP estimates do not take into account accuracy, i.e., any systematic location bias. The data is then first centered on the empirical group mean, assumed to coincide with the point of aim. The resulting CEP/SEP-MPI (around Mean Point of Impact) is a pure precision (spread) measure. Set accuracy=TRUE to incorporate systematic accuracy bias such that the point of aim is in the origin 0, possibly offset from the true group center.

- CorrNormal: For accuracy=FALSE and two-dimensional data, this estimate is based on the correlated bivariate normal distribution re-written in polar coordinates (radius and angle) (see Hoyt). For accuracy=TRUE or three-dimensional data, it is based on the (offset) circle/sphere probabilities for the correlated multivariate normal distribution (DiDonato & Jarnagin, 1961; DiDonato, 1981, see qmvnEll). This estimate is available for all probability levels.
- GrubbsPearson: The Grubbs-Pearson estimate (Grubbs, 1964) is based on the Pearson three-moment central chi^2-approximation of the true cumulative distribution function of radial error. The eigenvalues of the covariance matrix of shot-coordinates are used as variance estimates since they are the variances of the principal components (the PCA-rotated = decorrelated data). This estimate is available for all probability levels, and generalizes to three dimensions.
- GrubbsPatnaik: The Grubbs-Patnaik estimate (Grubbs, 1964) differs from the Grubbs-Pearson estimate insofar as it is based on the Patnaik two-moment central chi^2-approximation of the true cumulative distribution function of radial error.
- GrubbsLiu: The Grubbs-Liu estimate was not proposed by Grubbs but follows the same principle as his original estimates. It differs from them insofar as it is based on the Liu-Tang-Zhang four-moment non-central chi^2-approximation of the true cumulative distribution function of radial error. For accuracy=FALSE, it is identical to GrubbsPearson.
- Rayleigh: For accuracy=FALSE and two-dimensional data, this estimate uses the Rayleigh distribution (see getRayParam). It is valid for uncorrelated bivariate normal coordinates with equal variances. This estimate is available for all probability levels. For accuracy=FALSE and three-dimensional data, the Maxwell-Boltzmann distribution is used (see getRayParam). For accuracy=TRUE and two-dimensional data, the estimate uses the Rice distribution (see getRiceParam). For accuracy=TRUE and three-dimensional data, it is based on the offset sphere probabilities for the multivariate normal distribution set to have equal variances (see qmvnEll).
- Krempasky: The Krempasky estimate (Krempasky, 2003) is based on a nearly exact closed-form solution for the 50% quantile of the radial error for the correlated bivariate normal distribution with 0 mean. It requires estimating the covariance matrix and can only be reported for probability 0.5 with accuracy=FALSE. It does not generalize to three dimensions.
- Ignani: The Ignani estimate (Ignani, 2010) is based on a polynomial approximation for some quantiles of the radial error for the correlated bivariate normal distribution with 0 mean. It requires estimating the covariance matrix and can only be reported for probabilities 0.5, 0.9, 0.95, 0.99 with accuracy=FALSE. It generalizes to three dimensions.
- RMSE: For accuracy=FALSE, this estimator is the RMSE estimator often described in the GPS literature (van Diggelen, 2007) when using centered data for calculating RMSE (square root of the mean squared error). It is very similar to the Rayleigh estimator. For accuracy=TRUE, this the RMSE estimator often described in the GPS literature when using the original, non-centered data for calculating RMSE. It is similar to the Rayleigh estimator only when bias is small, but becomes seriously wrong otherwise. It is available for all probability levels, and generalizes to three dimensions.

- `Ethridge`: The Ethridge estimate (Ethridge, 1983; Puhek, 1992) is not based on the assumption of multivariate normality of coordinates but uses a robust unbiased estimator for the median radius (Hogg, 1967). It can only be reported for probability 0.5 but generalizes to three dimensions.
- `RAND`: The modified RAND R-234 estimate (RAND, 1952; Pesapane & Irvine, 1977; Puhek 1992) is a weighted sum of the square root of the eigenvalues of the covariance matrix of shot coordinates (the standard deviations of the data that is first de-correlated through rotation). It can only be reported for probability 0.5 and does not generalize to three dimensions.
- `Valstar`: Very similar to the RAND R-234 estimate with `accuracy=FALSE` except for very elliptical distributions but with a different bias correction with `accuracy=TRUE`. It can only be reported for probability 0.5 and does not generalize to three dimensions.

Estimators based on the normal distribution use the plug-in method (Blischke & Halpin, 1966), i.e., they substitute the true covariance matrix and mean vector with those estimated from the data. They are thus strictly valid only for the asymptotic distribution, while the finite sample distribution may differ somewhat.

If package `shiny` is installed, an interactive web app for this functionality can be run with `runGUI("hitprob")`.

Value

A list with the (chosen) CEP/SEP estimates and supplementary information on the distribution shape.

`CEP`	a numerical matrix with the chosen CEP/SEP estimates at the indicated `CEPlevel` (in original measurement units, MOA, SMOA, milliradian).
`ellShape`	a numerical vector with the aspect ratio of the ellipse (the square root of condition index `kappa`) and its flattening (1 - inverse aspect ratio).
`ctr`	group center

References

Blischke W. R. and Halpin, A. H. (1966). Asymptotic properties of some estimators of quantiles of circular error. Journal of the American Statistical Association, 61 (315), 618-632.

DiDonato, A. R. (1988). Integration of the trivariate normal distribution over an offset spehere and an inverse problem (NSWC TR 87-27). Dahlgren, VA: U.S. Naval Surface Weapons Center Dahlgren Division.

DiDonato, A. R., & Jarnagin, M. P. (1961). Integration of the general bivariate Gaussian distribution over an offset circle. Mathematics of Computation, 15 (76), 375-382.

Grubbs, F. E. (1964). Approximate circular and noncircular offset probabilities of hitting. Operations Research, 12(1), 51-62.

Hogg, R. V. (1967). Some observations on robust estimation. Journal of the American Statistical Association, 62 (320), 1179-1186.

Ignani, B. (2010). Determination of Circular and Spherical Position-Error Bounds in System Performance Analysis. Journal of Guidance, Control, and Dynamics, 33 (4), 1301-1304.

Krempasky, J. J. (2003). CEP equation exact to the fourth order. Navigation: Journal of The Institute of Navigation, 50 (3), 143-149.

Liu, H., Tang, Y., & Zhang, H. H. (2009). A new chi-square approximation to the distribution of non-negative definite quadratic forms in non-central normal variables. Computational Statistics & Data Analysis, 53(4), 853-856.

Pesapane, J., & Irvine, R. B. (1977). Derivation of CEP formula to approximate RAND-234 tables. Offut AFB, NE: Ballistic Missile Evaluation, HQ SAC.

Puhek, P. (1992). Sensitivity analysis of circular error probable approximation techniques (AFIT/GOR/ENS/92M-23). Wright-Patterson AFB, OH: Air Force Institute of Technology.

RAND Corporation. (1952). Offset circle probabilities (RAND-234). Santa Monica, CA: RAND Corporation.

Singh, H. P. 1992. Estimation of Circular Probable Error. The Indian Journal of Statistics, Series B 5(3), 289-305.

van Diggelen, F. 2007. Update: GNSS Accuracy: Lies, Damn Lies, and Statistics. GPS World.

See Also

Rayleigh, Maxwell, Hoyt, Rice, mvnEll, getHoytParam, getRayParam, getRiceParam, getConfEll, getHitProb, covMcd

Examples

```
# coordinates given by a suitable data frame
(cep <- getCEP(DFtalon, CEPlevel=0.5, accuracy=FALSE,
               dstTarget=10, conversion='m2mm',
               type=c('CorrNormal', 'GrubbsPatnaik', 'Rayleigh')))

# plot points, centers, and circles indicating 50%-CEP estimates
plot(point.y ~ point.x, data=DFtalon, asp=1, pch=16)
drawCircle(cep$ctr, cep$CEP$CEP0.5['unit', 'CorrNormal'],    fg='red')
drawCircle(cep$ctr, cep$CEP$CEP0.5['unit', 'GrubbsPatnaik'], fg='green3')
drawCircle(cep$ctr, cep$CEP$CEP0.5['unit', 'Rayleigh'],      fg='blue')
points(cep$ctr[1], cep$ctr[2], pch=4, col='gray50', cex=2, lwd=2)
legend(x='bottomleft',
       legend=c('Grubbs 50', 'Corr Normal 50', 'Rayleigh 50', 'center'),
       col=c('red', 'green3', 'blue', 'gray50'), lwd=2,
       lty=c(1, 1, 1, NA), pch=c(NA, NA, NA, 4), bg='white')

# calculate actual coverage percentage of 50% CEP estimates
dists <- getDistToCtr(DFtalon)           # distances to center

# extract CEP radius for all estimates
CEPr <- cep$CEP$CEP0.5['unit', ]

# percentage of points in circle with radius = CEP
100 * sapply(CEPr, function(x) sum(dists <= x)) / length(dists)

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(100, 0, 5), 2), ncol=2)
getCEP(xy, accuracy=FALSE, dstTarget=25, conversion='m2cm',
       type=c('Grubbs', 'CorrNormal', 'Rayleigh'))
```

```
## End(Not run)
```

`getConfEll` *Confidence ellipse*

Description

Calculates the confidence ellipse for the true mean of shot coordinates under the assumption of multivariate normality. Also includes the ellipse based on a robust estimate for the covariance matrix of the shot coordinates.

Usage

```
getConfEll(xy, level = 0.5, dstTarget, conversion,
          center = FALSE, doRob=TRUE)

## S3 method for class 'data.frame'
getConfEll(xy, level = 0.5, dstTarget, conversion,
          center = FALSE, doRob=TRUE)

## Default S3 method:
getConfEll(xy, level = 0.5, dstTarget, conversion,
          center = FALSE, doRob=TRUE)
```

Arguments

`xy`	Shot coordinates of n points: either a numerical (n x p)-matrix (1 row of p coordinates per point), or a data frame with either the variables `x`, `y` or `point.x`, `point.y` (optionally `z` or `point.z`).
`level`	a numerical value with the coverage for the confidence ellipse.
`dstTarget`	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable `distance` is already included in `xy`. See `getMOA`.
`conversion`	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables `dist.unit` and `point.unit` are already included in `xy`. Example `'m2cm'`. See `getMOA`.
`center`	logical: center groups to mean (0,0) first? If variable `series` does not exist, all shots are treated as belonging to the same group. Only available in method `getConfEll.data.frame()`.
`doRob`	logical: should robust covariance matrix estimate be used as well?

Details

No coordinate transforms are done (unlike in `groupLocation`, `groupShape`, `groupSpread`), i.e., origin is not assumed to be in top-left corner, and points are not taken relative to point of aim.

Robust estimate for the covariance matrix of coordinates is from `covMcd` using the MCD algorithm. See `getCEP` for estimates of the circular/spherical error probable.

Value

A list with the confidence ellipse measures.

ctr	coordinates group center.
ctrRob	coordinates robust estimate of group center.
cov	covariance matrix.
covRob	robust estimate of covariance matrix.
size	a numerical matrix with the lengths of the semi-axes of the ellipse (in original measurement units, MOA, SMOA, milliradian).
sizeRob	a numerical matrix with the lengths of the semi-axes of the ellipse based on a robust estimate for the covariance matrix of shot coordinates (in original measurement units, MOA, SMOA, milliradian).
shape	a numerical vector with the angle, the aspect ratio of the ellipse (square root of condition index kappa), its flattening (1 - inverse aspect ratio) as well as the trace and determinant of the covariance matrix.
shapeRob	a numerical vector with the aspect ratio and the flattening of the ellipse as well as the trace and determinant based on a robust estimate for the covariance matrix of shot coordinates.
magFac	magnification factor used to turn the error ellipse into the confidence ellipse as determined by the F(p, n-1)-distribution.

See Also

getCEP, covMcd, drawEllipse

Examples

```
# coordinates given by a suitable data frame
(ce <- getConfEll(DFsavage, level=0.5, dstTarget=100,
                  conversion='yd2in'))

# plot points, center, parametric confidence
# ellipse, and its axes
plot(point.y ~ point.x, data=DFsavage, asp=1, pch=16)
drawEllipse(ce, axes=TRUE, fg='blue', colCtr='blue', lwd=2, pch=4, cex=2)

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(100, 0, 5), 2), ncol=2)
getConfEll(xy, level=0.5, dstTarget=25, conversion='m2cm')

## End(Not run)
```

`getDistance` *Get distance based on absolute and angular size*

Description

Calculates the distance to an object based on the object's absolute and angular size.

Usage

```
getDistance(x, angular, conversion,
            type = c('deg', 'rad', 'MOA', 'SMOA', 'mrad', 'mil'))
```

Arguments

`x`	a numerical vector of absolute object sizes.
`angular`	a numerical vector of angular object sizes.
`conversion`	how to convert the measurement unit for the returned distance to object to that of given absolute object size. Either a character vector indicating the conversion such as `'m2cm'` for meter to centimeter, `'yd2in'` for yards to inches, or `'ft2cm'` for feet to cm. Imperial (yd, ft, in) and metric units (m, cm, mm) can be freely mixed. Alternatively, a numerical vector giving the multiplication factor for conversion: 100 for m to cm, 36 for yd to in, and 12 for ft to in.
`type`	type of angular measure used in `angular`: `'deg'` for degree, `'rad'` for radian, `'MOA'` for minute of angle, `'SMOA'` for Shooter's MOA, `'mrad'` for milliradian, `'mil'` for NATO mil. See details.

Details

1 MOA (minute of angle, arcmin) = 1/60 degree. Shooter's MOA = SMOA = Inches Per Hundred Yards IPHY. 1 inch at 100 yards = 1 SMOA. 1 milliradian = 1/1000 radian. 1 mil = 2*pi/6400 radian (NATO definition: the circle circumference is divided into 6400 mils). Details are given in the vignette, see `vignette('shotGroups')`

Value

A numerical vector with the distance values.

See Also

`getMOA`, `fromMOA`

Examples

```
size <- seq(1, 20, by=5)   # inch
dst  <- 100                # yard

# get angular size in MOA from absolute size
angular <- getMOA(size, dst=dst, conversion='yd2in', type='MOA')

# this should return dst throughout
getDistance(size, angular=angular, conversion='yd2in', type='MOA')
```

`getDistToCtr` *Distances to center for a set of points*

Description

Calculates the distances of a set of points to their center.

Usage

```
getDistToCtr(xy)

## S3 method for class 'data.frame'
getDistToCtr(xy)

## Default S3 method:
getDistToCtr(xy)
```

Arguments

`xy`	either a numerical (n x p)-matrix with the coordinates of n points in p-dimensional space (1 row of coordinates per point), or a data frame with either the variables `x`, `y` or `point.x`, `point.y`.

Value

A numerical vector with the distances from each point to the center of the set.

Examples

```
# coordinates given by a suitable data frame
getDistToCtr(DFtalon)

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(20, 100, 15), 1), ncol=2)
getDistToCtr(xy)

## End(Not run)
```

getHitProb *Hit probability within given region*

Description

Calculates the hit probability within a circular or spherical region for a given group of two- or three-dimensional coordinates.

Usage

```
getHitProb(xy, r=1, unit = 'unit', dstTarget, conversion,
           center = FALSE, accuracy = FALSE, type = 'CorrNormal', doRob = FALSE)

## S3 method for class 'data.frame'
getHitProb(xy, r=1, unit = 'unit', dstTarget, conversion,
           center = FALSE, accuracy = FALSE, type = 'CorrNormal', doRob = FALSE)

## Default S3 method:
getHitProb(xy, r=1, unit = 'unit', dstTarget, conversion,
           center = FALSE, accuracy = FALSE, type = 'CorrNormal', doRob = FALSE)
```

Arguments

xy	either a numerical (n x p)-matrix with the coordinates of n points (1 row of p coordinates per point), or a data frame with either the variables x, y or point.x, point.y (optionally z or point.z).
r	a numerical vector with the radius values for the circle/sphere that defines the region for which the probability should be calculated.
unit	Measurement unit for radius r. Default 'unit' indicates that the measurement unit for (x,y)-coordinates given in conversion. Possible values are 'unit', 'm', 'cm', 'mm', 'yd', 'ft', 'in', 'deg', 'MOA', 'SMOA', 'rad', 'mrad', 'mil'.
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in xy. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables dist.unit and point.unit are already included in xy. Example 'm2cm'. See getMOA.
center	logical: center groups to mean (0,0) first? If variable series does not exist, all shots are treated as belonging to the same group. Only available in method getHitProb.data.frame().
accuracy	logical: take systematic location bias into account? See details.
type	string vector indicating which estimation method to use. Possible values are 'CorrNormal', 'GrubbsPearson', 'GrubbsPatnaik', 'GrubbsLiu', 'Rayleigh'.
doRob	logical: use robust estimation of center and covariance matrix as basis for estimates? For Rayleigh only available when accuracy=FALSE.

Details

The estimated hit probability is the same as the estimated fraction of shots with a distance to their group center of at most `r`.

For `accuracy=FALSE` (default), the estimated hit probability does not take into account accuracy, i.e., any systematic location bias. The data is then first centered on the empirical group mean, assumed to coincide with the point of aim. Set `accuracy=TRUE` to incorporate systematic accuracy bias such that the point of aim is in the origin 0, possibly offset from the true group center.

- `CorrNormal`: For `accuracy=FALSE` and two-dimensional data, this estimate is based on the quantile function of the correlated bivariate normal distribution re-written in polar coordinates (radius and angle) (see `Hoyt`). For `accuracy=TRUE` or three-dimensional data, it is based on the (offset) circle/sphere probabilities for the correlated multivariate normal distribution (DiDonato & Jarnagin, 1961; DiDonato, 1981, see `pmvnEll`).
- `GrubbsPearson`: The Grubbs-Pearson estimate (Grubbs, 1964) is based on the Pearson three-moment central chi^2-approximation of the true cumulative distribution function of radial error. The eigenvalues of the covariance matrix of shot-coordinates are used as variance estimates since they are the variances of the principal components (the PCA-rotated = decorrelated data).
- `GrubbsPatnaik`: The Grubbs-Patnaik estimate (Grubbs, 1964) differs from the Grubbs-Pearson estimate insofar as it is based on the Patnaik two-moment central chi^2-approximation of the true cumulative distribution function of radial error.
- `GrubbsLiu`: The Grubbs-Liu estimate was not proposed by Grubbs but follows the same principle as his original estimates. It differs from them insofar as it is based on the Liu-Tang-Zhang four-moment non-central chi^2-approximation of the true cumulative distribution function of radial error. For `accuracy=FALSE`, it is identical to `GrubbsPearson`.
- `Rayleigh`: For `accuracy=FALSE` and two-dimensional data, this estimate uses the Rayleigh distribution (see `getRayParam`). It is valid for uncorrelated bivariate normal coordinates with equal variances. This estimate is available for all probability levels. For `accuracy=FALSE` and three-dimensional data, the Maxwell-Boltzmann distribution is used (see `getRayParam`). For `accuracy=TRUE` and two-dimensional data, the estimate uses the Rice distribution (see `getRiceParam`). For `accuracy=TRUE` and three-dimensional data, it is based on the offset sphere probabilities for the multivariate normal distribution set to have equal variances (see `qmvnEll`).

If package `shiny` is installed, an interactive web app for this functionality can be run with `runGUI("hitprob")`.

Value

A vector with the (chosen) hit-probability estimates. For more than one `r` and more than one `type`, a matrix.

References

DiDonato, A. R. (1988). Integration of the trivariate normal distribution over an offset spehere and an inverse problem (NSWC TR 87-27). Dahlgren, VA: U.S. Naval Surface Weapons Center Dahlgren Division.

DiDonato, A. R., & Jarnagin, M. P. (1961). Integration of the general bivariate Gaussian distribution over an offset circle. Mathematics of Computation, 15 (76), 375-382.

Grubbs, F. E. (1964). Approximate circular and noncircular offset probabilities of hitting. Operations Research, 12(1), 51-62.

Liu, H., Tang, Y., & Zhang, H. H. (2009). A new chi-square approximation to the distribution of non-negative definite quadratic forms in non-central normal variables. Computational Statistics & Data Analysis, 53(4), 853-856.

Singh, H. P. 1992. Estimation of Circular Probable Error. The Indian Journal of Statistics, Series B 5(3), 289-305.

See Also

Rayleigh, Maxwell, Hoyt, mvnEll, getHoytParam, getRayParam, getCEP, getConfEll, covMcd

Examples

```
# coordinates given by a suitable data frame
# estimated fraction of shots within a circle with radius
# 1 and 1.5 MOA.
getHitProb(DFscar17, r=c(1, 1.5), unit='MOA', accuracy=FALSE,
           dstTarget=100, conversion='yd2in',
           type=c('CorrNormal', 'GrubbsPatnaik'))

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(100, 0, 5), 2), ncol=2)
getHitProb(xy, r=c(2, 2.5), unit='MOA', accuracy=FALSE,
           dstTarget=100, conversion='yd2in',
           type=c('CorrNormal', 'GrubbsPatnaik'))

## End(Not run)
```

---

getHoytParam *Determine parameters q and omega of the Hoyt distribution*

Description

Determines the Hoyt distribution's shape parameter q and scale parameter omega from the eigenvalues of a (2 x 2)-covariance matrix.

Usage

```
getHoytParam(x)

## S3 method for class 'matrix'
getHoytParam(x)

## S3 method for class 'list'
getHoytParam(x)
```

```
## S3 method for class 'data.frame'
getHoytParam(x)

## Default S3 method:
getHoytParam(x)
```

Arguments

x	one of the following: a (2 x 2)-covariance matrix, a list of (2 x 2)-covariance matrices, a data frame with either the variables x, y or point.x, point.y, a 2-vector with eigenvalues.

Details

The parameters q and omega derive from the eigenvalues ev1, ev2 of the covariance matrix of the bivariate normal distribution as follows: `q = 1 / sqrt(((ev1+ev2)/ev2) - 1)` and `omega = ev1 + ev2`.

If x is a data frame, its sample covariance matrix is used to estimate the eigenvalues. Note that the Hoyt distribution is only approximately valid for large samples if estimated parameters are used.

Value

A list with the following components:

q	A vector with values of the shape parameter q.
omega	A vector with values of the scale parameter omega.

References

Hoyt, R. S. (1947). Probability functions for the modulus and angle of the normal complex variate. Bell System Technical Journal, 26(2), 318-359.

https://reference.wolfram.com/language/ref/HoytDistribution.html

See Also

Hoyt

Examples

```
## q and omega based on coordinates in a data frame
getHoytParam(DFscar17)

## q and omega based on a covariance matrix
cm1 <- cbind(c(8, 0), c(0, 2))
getHoytParam(cm1)

## q and omega based on a list of covariance matrices
cm2 <- cbind(c(6, 0), c(0, 4))
cmL <- list(cm1, cm2)
getHoytParam(cmL)
```

```
## q and omega based on eigenvalues
ev <- eigen(cm1)$values
getHoytParam(cm1)
```

`getKuchnost` *Kuchnost precision estimate*

Description

Estimates the Kuchnost precision measure. The maximum distance to the group center after removing outliers.

Usage

```
getKuchnost(xy, dstTarget, conversion,
       center = FALSE, doRob = FALSE, strict = FALSE)

## S3 method for class 'data.frame'
getKuchnost(xy, dstTarget, conversion,
       center = FALSE, doRob = FALSE, strict = FALSE)

## Default S3 method:
getKuchnost(xy, dstTarget, conversion,
       center = FALSE, doRob = FALSE, strict = FALSE)
```

Arguments

`xy`	either a numerical (n x p)-matrix with the coordinates of n points (1 row of p coordinates per point), or a data frame with either the variables `x`, `y` or `point.x`, `point.y` (optionally `z` or `point.z`).
`dstTarget`	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable `distance` is already included in `xy`. See `getMOA`.
`conversion`	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables `dist.unit` and `point.unit` are already included in `xy`. Example `'m2cm'`. See `getMOA`.
`center`	logical: center groups to mean (0,0) first? If variable `series` does not exist, all shots are treated as belonging to the same group. Only available in method `getCEP.data.frame()`.
`doRob`	logical: use robust estimation of center and covariance matrix as basis for estimators?
`strict`	logical: enable check that `xy` contains exactly 4 shots.

Details

An outlier is defined as a shot that is at least 2.5 times further away from the group center defined by all remaining shots compared to the maximum distance to center of those shots themselves.

Value

A list with 3 components.

Kuchnost	The Kuchnost precision measure.
outlier	Indices of outliers, if any.
ctr	group center after removing outliers

References

https://github.com/lstange/mcgs

See Also

covMcd

Examples

```
getKuchnost(DF300BLK)
```

---

getMaxPairDist *Maximum pairwise distance for a set of points*

Description

Calculates the maximum of pairwise distances between points given a set of coordinates.

Usage

```
getMaxPairDist(xy)

## S3 method for class 'data.frame'
getMaxPairDist(xy)

## Default S3 method:
getMaxPairDist(xy)
```

Arguments

xy	either a numerical (n x p)-matrix with the coordinates of n points in p-dimensional space (1 row of coordinates per point), or a data frame with either the variables x, y or point.x, point.y.

Value

A list with the following components

d	numerical value with the maximum pairwise distance between points.
idx	a vector with two entries giving the row indices of the points that are farthest apart.

Examples

```
# coordinates given by a suitable data frame
(maxPD <- getMaxPairDist(DFsavage))

# plot points and point pair with maximum distance
plot(point.y ~ point.x, data=DFsavage, asp=1, pch=16)
x0 <- DFsavage$point.x[maxPD$idx[1]]     # 1st point x
y0 <- DFsavage$point.y[maxPD$idx[1]]     # 1st point y
x1 <- DFsavage$point.x[maxPD$idx[2]]     # 2nd point x
y1 <- DFsavage$point.y[maxPD$idx[2]]     # 2nd point y
segments(x0, y0, x1, y1, col="green3", lwd=2)

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(20, 100, 15), 1), ncol=2)
getMaxPairDist(xy)

## End(Not run)
```

---

getMinBBox *Minimum-area bounding box for a set of 2D-points*

Description

Calculates the vertices of the minimum-area, possibly oriented bounding box given a set of 2D-coordinates.

Usage

```
getMinBBox(xy)

## S3 method for class 'data.frame'
getMinBBox(xy)

## Default S3 method:
getMinBBox(xy)
```

Arguments

xy	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n >= 2 points (1 row of coordinates per point), or a data frame with either the variables x, y or point.x, point.y.

Details

No coordinate transforms are done (unlike in groupLocation, groupShape, groupSpread), i.e., origin is not assumed to be in top-left corner, and points are not taken relative to point of aim.
Uses the rotating calipers algorithm (Toussaint, 1983).

Value

A list with the following information about the minimum-area bounding box:

`pts`	a (4 x 2)-matrix containing the coordinates of the (ordered) vertices.
`width`	width of the box.
`height`	height of the box.
`FoM`	figure of merit, i.e., the average side length of the box: (`width` + `height`) / 2.
`diag`	length of box diagonal.
`angle`	orientation of the box' longer edge pointing up as returned by `atan2`, but in degree.

References

Computational Geometry Algorithms Library. 2021. CGAL Chapter 65: Bounding Volumes. https://doc.cgal.org/Manual/latest/doc_html/cgal_manual/Bounding_volumes/Chapter_main.html Toussaint, G. T. 1983. Solving geometric problems with the rotating calipers. In: Proceedings of the 1983 IEEE MELECON. Athens, Greece: IEEE Computer Society.

See Also

`drawBox2`, `getBoundingBox`, `getMinCircle`

Examples

```
# coordinates given by a suitable data frame
bb <- getMinBBox(DFsavage)              # minimum bounding box

# plot points and minimum bounding box
plot(point.y ~ point.x, data=DFsavage, asp=1,
     xlim=range(bb$pts[ , 1]), ylim=range(bb$pts[ , 2]), pch=16)
drawBox2(bb, fg='blue', colCtr='blue', pch=4, cex=2)

bb$FoM                                  # figure of merit
bb$angle                                # box orientation

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(16, 100, 15)), ncol=2)
getMinBBox(xy)

## End(Not run)
```

`getMinCircle` *Minimum enclosing circle for a set of 2D-points*

Description

Calculates center and radius of the minimum enclosing circle given a set of 2D-coordinates.

Usage

```
getMinCircle(xy)

## S3 method for class 'data.frame'
getMinCircle(xy)

## Default S3 method:
getMinCircle(xy)
```

Arguments

`xy`	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n >= 2 points (1 row of coordinates per point), or a data frame with either the variables `x`, `y` or `point.x`, `point.y`.

Details

No coordinate transforms are done (unlike in `groupLocation`, `groupShape`, `groupSpread`), i.e., origin is not assumed to be in top-left corner, and points are not taken relative to point of aim.

Uses the Skyum algorithm based on the convex hull.

Value

A list containing the center and radius of the circle.

`ctr`	a numerical 2-vector giving the (x,y)-coordinates of the circle's center.
`rad`	a numerical value giving the radius of the circle.

References

Computational Geometry Algorithms Library. 2021. CGAL Chapter 65: Bounding Volumes. https://doc.cgal.org/Manual/latest/doc_html/cgal_manual/Bounding_volumes/Chapter_main.html Fischer, K.; Gaertner, B.; Kutz, M. 2003. Fast smallest-enclosing-ball computation in high dimensions. In: Proceedings of the 11th European Symposium on Algorithms (ESA), 630-641. https://github.com/hbf/miniball Gaertner, B. 2021. Miniball: Smallest Enclosing Balls of Points. https://people.inf.ethz.ch/gaertner/subdir/software/miniball.html Skyum, S. 1991. A simple algorithm for computing the smallest enclosing circle. Information Processing Letters 37(3), 121-125. Welzl, E. 1991. Smallest enclosing disks (balls and ellipsoids). In: Maurer H. (eds), New Results and New Trends in Computer Science 555, 359-370. doi 10.1007/BFb0038202.

See Also

drawCircle, getMinBBox, getBoundingBox

Examples

```
# coordinates given by a suitable data frame
mc <- getMinCircle(DFsavage)

# determine axis limits so that circle will be visible
xLims <- mc$ctr[1] + c(-mc$rad, mc$rad)
yLims <- mc$ctr[2] + c(-mc$rad, mc$rad)
plot(point.y ~ point.x, data=DFsavage,
     pch=16, asp=1, xlim=xLims, ylim=yLims)
drawCircle(mc, fg='blue')

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(20, 100, 15), 2), ncol=2)
getMinCircle(xy)

## End(Not run)
```

---

getMinEllipse *Minimum enclosing ellipse for a set of 2D-points*

Description

Calculates center, shape matrix, and area of the minimum enclosing ellipse given a set of 2D-coordinates using Khachiyan's algorithm.

Usage

```
getMinEllipse(xy, tol = 0.001, max_iter = 1000)

## S3 method for class 'data.frame'
getMinEllipse(xy, tol = 0.001, max_iter = 1000)

## Default S3 method:
getMinEllipse(xy, tol = 0.001, max_iter = 1000)
```

Arguments

xy	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n >= 2 points (1 row of coordinates per point), or a data frame with either the variables x, y or point.x, point.y.
tol	numerical tolerance value. Should be < 1.
max_iter	maximum number of iterations.

Details

No coordinate transforms are done (unlike in groupLocation, groupShape, groupSpread), i.e., origin is not assumed to be in top-left corner, and points are not taken relative to point of aim.

Value

A list containing the center, (2 x 2)-shape matrix, and area of the ellipse.

ctr	a numerical 2-vector giving the (x,y)-coordinates of the ellipse's center.
E	a numerical positive definite (2 x 2)-matrix defining the ellipse in the form ((x-c)' E (x-c)) <= 1
cov	matrix, whose eigen-structure determines shape of ellipse. Inverse of E. Useful for plotting with drawEllipse.
area	a numerical value giving the area of the ellipse.
shape	a numerical vector with the orientation of the ellipse's major axis pointing up as returned by atan2 (but in degree), the aspect ratio of the ellipse (square root of condition index kappa), its flattening (1 - inverse aspect ratio) as well as the trace and determinant of the covariance matrix.
size	a numerical vector with the lengths of the semi-axes of the ellipse.

References

Computational Geometry Algorithms Library. 2021. CGAL Chapter 65: Bounding Volumes. https://doc.cgal.org/Manual/latest/doc_html/cgal_manual/Bounding_volumes/Chapter_main.html

Todd MJ and Yildrim EA. On Khachiyan's Algorithm for the Computation of Minimum Volume Enclosing Ellipsoids. Discrete Applied Mathematics 2007: 155: 1731-1744. https://people.orie.cornell.edu/miketodd/TYKhach.pdf

Jacob. Bounding ellipse. https://stackoverflow.com/a/1768440.

See Also

drawEllipse, getMinCircle, getMinBBox, getBoundingBox

Examples

```
# coordinates given by a suitable data frame
me <- getMinEllipse(DFsavage, tol=0.001)

plot(point.y ~ point.x, data=DFsavage, pch=16, asp=1)
drawEllipse(me, fg='blue')

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(20, 100, 15), 2), ncol=2)
getMinEllipse(xy, tol=0.001)

## End(Not run)
```

getMOA *Conversion of absolute size to angular diameter*

Description

Converts object size to angular diameter (degree, radian, minute of angle MOA = arcminute, Shooter's MOA SMOA, milliradian mrad, NATO mil).

Usage

```
getMOA(x, dst, conversion,
       type = c('deg', 'rad', 'MOA', 'SMOA', 'mrad', 'mil'))
```

Arguments

x	a numerical vector of object sizes.
dst	a numerical vector of viewing distances.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates. Either a character vector indicating the conversion such as 'm2cm' for meter to centimeter, 'yd2in' for yards to inches, or 'ft2cm' for feet to cm. Imperial (yd, ft, in) and metric units (m, cm, mm) can be freely mixed. Alternatively, a numerical vector giving the multiplication factor for conversion: 100 for m to cm, 36 for yd to in, and 12 for ft to in.
type	type of angular diameter: 'deg' for degree, 'rad' for radian, 'MOA' for minute of angle, 'SMOA' for Shooter's MOA, 'mrad' for milliradian, 'mil' for NATO mil. See details.

Details

1 MOA (minute of angle, arcmin) = 1/60 degree. Shooter's MOA = SMOA = Inches Per Hundred Yards IPHY. 1 inch at 100 yards = 1 SMOA. 1 milliradian = 1/1000 radian. 1 mil = 2*pi/6400 radian (NATO definition: the circle circumference is divided into 6400 mils). Details are given in the vignette, see `vignette('shotGroups')` .

If package shiny is installed, an interactive web app for this functionality can be run with `runGUI("angular")`.

Value

A numerical vector with the angular diameter values.

See Also

fromMOA, getDistance

Examples

```
size <- seq(1, 20, by=5)   # inch
dst  <- 100                # yard
getMOA(size, dst=dst, conversion='yd2in', type='MOA')

# this should return objSize
MOA <- getMOA(size, dst=dst, conversion='yd2in', type='MOA')
fromMOA(MOA, dst=dst, conversion='yd2in', type='MOA')

# SMOA
getMOA(c(1, 2, 5), dst=100, conversion='yd2in', type='SMOA')

# milliradian
getMOA(c(10, 20, 50), dst=100, conversion='m2mm', type='mrad')
```

---

getRangeStat *Range statistics*

Description

Returns range statistics: extreme spread, figure of merit, bounding box diagonal

Usage

```
getRangeStat(xy, dstTarget, conversion)

## S3 method for class 'data.frame'
getRangeStat(xy, dstTarget, conversion)

## Default S3 method:
getRangeStat(xy, dstTarget, conversion)
```

Arguments

xy	either a numerical (n x p)-matrix with the coordinates of n points (1 row of p coordinates per point), or a data frame with either the variables x, y or point.x, point.y (optionally z or point.z).
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in xy. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables dist.unit and point.unit are already included in xy. Example 'm2cm'. See getMOA.

Value

A named numeric vector with elements ES (extreme spread), FoM (figure of merit), D (bounding box diagonal).

References

Grubbs, F. E. (1964b). Statistical measures of accuracy for riflemen and missile engineers. Ann Arbor, ML: Edwards Brothers.

See Also

getMaxPairDist, getBoundingBox

Examples

```
getRangeStat(DFscar17)
```

`getRayParam` *Estimate Rayleigh parameters sigma, mean and standard deviation*

Description

Estimates the radial precision parameter sigma of the Rayleigh distribution together with the radial mean MR and radial standard deviation RSD, including parametric confidence intervals. For 1D data, it estimates the parameters of the half normal distribution. For 3D data, it estimates the parameters of the Maxwell-Boltzmann distribution

Usage

```
getRayParam(xy, level = 0.95, mu, doRob = FALSE)

## S3 method for class 'data.frame'
getRayParam(xy, level = 0.95, mu, doRob = FALSE)

## Default S3 method:
getRayParam(xy, level = 0.95, mu, doRob = FALSE)
```

Arguments

`xy`	either a numerical matrix with the coordinates of n points (1 row of coordinates per point), or a data frame with either the variables `x`, `y` or `point.x`, `point.y`.
`level`	a numerical value with the coverage for the confidence intervals for sigma, MR, RSD.
`mu`	numerical 2-vector with the true group center (optional). See details.
`doRob`	logical: use robust estimation of covariance matrix as basis for estimators?

Details

When the true mean mu of the distribution is given, the sigma estimate uses the sum of squared radii for the variance estimate (the total un-corrected variance of the coordinates), and employs the c4 correction factor for taking the square root. When mu is missing, the sum of squared radii is Bessel-corrected for estimating the center.

The robust estimate for the covariance matrix of (x,y)-coordinates is from covMcd using the MCD algorithm.

Value

A list with the estimates for sigma, RSD, and MR including the confidence intervals.

sigma	A vector with the sigma estimate and confidence interval bounds as named elements sigma, sigCIlo, sigCIup.
RSD	A vector with the RSD estimate and confidence interval bounds as named elements RSD, RSDciLo, RSDciUp.
MR	A vector with the MR estimate and confidence interval bounds as named elements MR, MRciLo, MRciUp.

References

http://ballistipedia.com/index.php?title=Closed_Form_Precision

Singh, H. P. 1992. Estimation of Circular Probable Error. The Indian Journal of Statistics, Series B 5(3), 289-305.

See Also

Rayleigh, Maxwell, getCEP, getHitProb, groupSpread, covMcd

Examples

```
# coordinates given by a suitable data frame
getRayParam(DFtalon, level=0.95, doRob=FALSE)

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(100, 0, 5), 2), ncol=2)
getRayParam(xy, level=0.95, doRob=FALSE)

## End(Not run)
```

getRiceParam *Estimate Rice parameters nu and sigma*

Description

Estimates the location parameter nu and the scale parameter sigma of the Rice distribution together with the radial mean MR and radial standard deviation RSD based on a set of 2D-coordinates. Includes the parametric confidence interval for sigma.

Usage

```
getRiceParam(xy, level = 0.95, doRob = FALSE, type = c('LiZhangDai', 'MOM'))

## S3 method for class 'data.frame'
getRiceParam(xy, level = 0.95, doRob = FALSE, type=c('LiZhangDai', 'MOM'))

## Default S3 method:
getRiceParam(xy, level = 0.95, doRob = FALSE, type=c("LiZhangDai", 'MOM'))
```

Arguments

xy	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n points (1 row of coordinates per point), or a data frame with either the variables x, y or point.x, point.y.
level	a numerical value with the coverage for the confidence interval for sigma.
doRob	logical: use robust estimation of center and covariance matrix as basis for estimators?
type	The initial biased nu estimate is the Euclidean norm of the group center. For 'LiZhangDai', the bias-correction from Liu et al., 2009. For 'MOM', the estimated bias is subtracted. If the estimated bias is larger than the initial estimate, the final estimate is then set to 0.

Details

The sigma estimate uses getRayParam. The robust estimate for the center and for the covariance matrix of (x,y,z)-coordinates is from covMcd using the MCD algorithm.

Value

A list with the estimates for nu, sigma, RSD, and MR including the confidence interval for sigma.

nu	The estimated location parameter nu.
sigma	A vector with the sigma estimate and confidence interval bounds as named elements sigma, sigCIlo, sigCIup.
MR	The MR estimate.
RSD	The RSD estimate.

References

https://reference.wolfram.com/language/ref/RiceDistribution.html

Li, Q., Zhang, J., & Dai, S. (2009). On estimating the non-centrality parameter of a chi-squared distribution. Statistics and Probability Letters 79, 98-114.

See Also

Rice, getRayParam, getCEP, getHitProb, covMcd

Examples

```
getRiceParam(DF300BLKhl, level=0.95, doRob=FALSE)

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(60, 0, 5), 2), ncol=2)
getRiceParam(xy, level=0.95, doRob=FALSE)

## End(Not run)
```

---

getXYmat *Extract (x,y)-coordinates (relative to point of aim) from a data frame*

Description

Extracts (x,y)- or (x,y,z)-coordinates of the points of impact (relative to the point of aim) from a data frame and returns them as a matrix.

Usage

```
getXYmat(DF, xyTopLeft = TRUE, relPOA = TRUE, center = FALSE)
```

Arguments

DF	a data frame containing (at least) either the variables point.x, point.y or x, y defining the bullet holes. For three-dimensional data, variables point.z or z are recognized. Variables aim.x, aim.y, and for three-dimensional data point.z can define the point of aim. If missing, point of aim is assumed to be in the origin 0.
xyTopLeft	logical: is the origin of the absolute coordinate system in the top-left corner? See details.
relPOA	logical: should returned coordinates be relative to the point of aim?
center	logical: center groups to mean (0,0)? If variable series does not exist, all shots are treated as belonging to the same group.

Details

By default, OnTarget PC/TDS' 'Export Point Data' places the origin of the absolute coordinate system in the top-left corner. In OnTarget TDS, this setting can be changed by checking the box 'Tools -> Options -> Options tab -> Data Export -> Invert Y-Axis on Export'. In that case, use `xyTopLeft=FALSE`. If groups appear to be upside-down, `xyTopLeft` is the setting to change.

Value

A numerical matrix with the (x,y)- or (x,y,z)-coordinates.

See Also

`groupLocation`, `groupShape`, `groupSpread`

Examples

```
data(DFcm)

# select data from only first series
DFsub <- subset(DFcm, series == 1)
getXYmat(DFsub, xyTopLeft=TRUE, relPOA=TRUE)
```

---

groupLocation *Accuracy: Location measures for a single group of bullet holes*

Description

Calculates location measures for a single group of bullet holes.

Usage

```
groupLocation(xy, level = 0.95, plots = TRUE, bootCI = 'none',
              dstTarget, conversion)

## S3 method for class 'data.frame'
groupLocation(xy, level = 0.95, plots = TRUE, bootCI = 'none',
              dstTarget, conversion)

## Default S3 method:
groupLocation(xy, level = 0.95, plots = TRUE, bootCI = 'none',
              dstTarget, conversion)
```

Arguments

xy	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n points (1 row of coordinates per point), or a data frame with either the variables x, y or point.x, point.y as well as aim.x, aim.y giving the point of aim. If missing, point of aim is assumed to be in (0,0).
level	a numerical value giving the level for the confidence intervals for the center (x,y)-coordinates.
plots	logical: show a 2D-scatterplot?
bootCI	a character vector to select which bootstrap confidence interval type to report. Possible types are 'none' (no bootstrap CI), 'norm', 'basic', 'perc', 'bca'. See boot.ci.
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in xy. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables dist.unit and point.unit are already included in xy. Example 'm2cm'. See getMOA.

Details

The number of replicates for the reported bootstrap confidence intervals is at least 1499. If the BCa interval is reported, it is at least the number of points.

If package shiny is installed, an interactive web app for this functionality can be run with runGUI("analyze").

Value

A list with the results from the numerical analyses and statistical tests.

ctr	(x,y)-offset of group center relative to point of aim.
ctrXci	parametric (t) and bootstrap confidence intervals for center x-coordinate.
ctrYci	parametric (t) and bootstrap confidence intervals for center y-coordinate.
ctrRob	robust estimate of group center offset relative to point of aim (MCD algorithm).
distPOA	distance from group center to point of aim (in original measurement units, MOA, SMOA, milliradian).
distPOArob	distance from robust estimate of group center to point of aim (in original measurement units, MOA, SMOA, milliradian).
Hotelling	Hotelling's T^2-Test result from testing if group center equals point of aim.

See Also

getMOA, covMcd, anova.mlm, boot, boot.ci

Examples

```
# coordinates given by a suitable data frame
res <- groupLocation(DFsavage, dstTarget=100, conversion='m2mm',
                     level=0.95, plots=2, bootCI='basic')
names(res)
res$ctr
res$distPOA
res$ctrXci
res$ctrYci

# coordinates given by a matrix
## Not run:
# assume data from pistol shooting at 25m with 9mm ammo
# metric units
xy <- matrix(round(rnorm(100, 0, 5), 2), ncol=2)
groupLocation(xy, dstTarget=25, conversion='m2cm', plots=2)

## End(Not run)
```

---

groupShape *Shape analysis for a single group of bullet holes*

Description

Assesses shape of a single group of bullet holes: Outlier analysis as well as numerical and graphical normality checks for a set of (x,y)-coordinates.

Usage

```
groupShape(xy, center = FALSE, plots = TRUE, bandW = 0.5,
           outlier = c('mcd', 'pca'), dstTarget, conversion, ...)

## S3 method for class 'data.frame'
groupShape(xy, center = FALSE, plots = TRUE, bandW = 0.5,
           outlier=c('mcd', 'pca'), dstTarget, conversion, ...)

## Default S3 method:
groupShape(xy, center = FALSE, plots = TRUE, bandW = 0.5,
           outlier=c('mcd', 'pca'), dstTarget, conversion, ...)
```

Arguments

xy	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n points (1 row of coordinates per point), or a data frame with either the variables `x`, `y` or `point.x`, `point.y` as well as `aim.x`, `aim.y` giving the point of aim. If missing, point of aim is assumed to be in (0,0).

center	logical: center groups to mean (0,0) first? If variable series does not exist, all shots are treated as belonging to the same group. Only available in method groupShape.data.frame().
plots	logical: show diagrams?
bandW	for argument bandwith of smoothScatter.
outlier	method for outlier identification: mcd uses robust Mahalanobis distances (see aq.plot), pca uses robust principal components analysis (see pcout). Requires installing package mvoutlier.
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in xy. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables dist.unit and point.unit are already included in xy. Example 'm2cm'. See getMOA.
...	additional arguments passed to pcout with outlier='pca' - final sensitivity can be adjusted with option outbound, a sensible candidate value seems to be around 0.45.

Details

In addition to the numerical results listed below, this function produces the following diagrams:

- a combined plot for multivariate outlier identification as produced by aq.plot - requires installing package mvoutlier
- a chi-square Q-Q-plot for eyeballing multivariate normality as produced by chisq.plot, including a reference line with intercept 0 and slope 1
- a heatmap of a non-parametric 2D-kernel density estimate for the (x,y)-coordinates as produced by smoothScatter together with group center and error ellipses (original and scaled by factor 2) based on a robust estimate for the covariance matrix (from covMcd using the MCD algorithm)
- a Q-Q-plot of x-coordinates for eyeballing normality
- a Q-Q-plot of y-coordinates for eyeballing normality
- a histogram of x-coordinates including a fitted normal distribution as well as a non-parametric kernel density estimate
- a histogram of y-coordinates including a fitted normal distribution as well as a non-parametric kernel density estimate

If package shiny is installed, an interactive web app for this functionality can be run with runGUI("analyze").

Value

A list with the results from the numerical analyses and statistical tests.

corXY	correlation matrix of (x,y)-coordinates.
corXYrob	robust estimate of correlation matrix of (x,y)-coordinates.
Outliers	a vector of row indices for observations identified as outliers - only if package mvoutlier is installed.

ShapiroX	Shapiro-Wilk-Test result for normality of x-coordinates. Only for at most 5000 points. For more than 5000 points, replaced by Kolmogorov-Smirnov-Test in ksX.
ShapiroY	Shapiro-Wilk-Test result for normality of y-coordinates. Only for at most 5000 points. For more than 5000 points, replaced by Kolmogorov-Smirnov-Test in ksY.
multNorm	E-statistic-Test result for multivariate normality of (x,y)-coordinates - only available if package energy is installed.

Note

The chi-square distribution is only strictly valid for squared Mahalanobis distances if the true center and the true covariance matrix are used in calculation. The goodness of approximation for situations where sample estimates are used should be sufficient here.

See Also

qqnorm, smoothScatter, hist, kernel, covMcd, shapiro.test, ks.test, mvnorm.etest, chisq.plot, aq.plot, pcout

Examples

```
# coordinates given by a suitable data frame
res <- groupShape(DFsavage, bandW=4, outlier='mcd',
                  dstTarget=100, conversion='m2mm')
names(res)
res$corXY
res$Outliers
res$multNorm

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(200, 0, 5), 2), ncol=2)
groupShape(xy, bandW=1.6)

## End(Not run)
```

---

groupSpread *Precision: Spread measures of a single group of bullet holes*

Description

Provides spread measures and their graphical representations for a single group of bullet holes.

Usage

```
groupSpread(xy, center = FALSE, plots = TRUE, CEPlevel = 0.5,
            CIlevel = 0.95, CEPtype = 'CorrNormal', bootCI = 'none',
            dstTarget, conversion)

## S3 method for class 'data.frame'
groupSpread(xy, center = FALSE, plots = TRUE, CEPlevel = 0.5,
            CIlevel = 0.95, CEPtype = 'CorrNormal', bootCI = 'none',
            dstTarget, conversion)

## Default S3 method:
groupSpread(xy, center = FALSE, plots = TRUE, CEPlevel = 0.5,
            CIlevel = 0.95, CEPtype = 'CorrNormal', bootCI = 'none',
            dstTarget, conversion)
```

Arguments

xy	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n points (1 row of coordinates per point), or a data frame with either the variables x, y or point.x, point.y as well as aim.x, aim.y giving the point of aim. If missing, point of aim is assumed to be in (0,0).
center	logical: center groups to mean (0,0) first? If variable series does not exist, all shots are treated as belonging to the same group. Only available in method groupSpread.data.frame().
plots	logical: show diagrams?
CEPlevel	a numerical value giving the coverage for the CEP and for the confidence ellipse.
CIlevel	a numerical value giving the level for the confidence intervals (for standard deviations as well as for Rayleigh sigma, RSD, MR).
CEPtype	string indicating which CEP estimate to report from getCEP.
bootCI	a character vector to select which bootstrap confidence interval type to report. Possible types are 'none' (no bootstrap CI), 'norm', 'basic', 'perc', 'bca'. See boot.ci.
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in xy. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Acts as override if variables dist.unit and point.unit are already included in xy. Example 'm2cm'. See getMOA.

Details

Explanations and formula for many reported precision measures such as CEP, sigma, RSD, MR, FoM can be found in the references.
Robust estimate for the covariance matrix of (x,y)-coordinates is from covMcd using the MCD algorithm.
The number of replicates for the reported bootstrap confidence intervals is at least 1499. If the BCa interval is reported, it is at least the number of points.
In addition to the numerical results listed below, this function produces the following diagrams:

- a scatterplot of the (x,y)-coordinates together with group center, circle with average distance to center, and 100*level%-confidence ellipse - the latter also based on a robust estimate for the covariance matrix
- a scatterplot of the (x,y)-coordinates together with the bounding box, minimum bounding box, minimum enclosing circle, and maximum group spread
- a histogram of distances to group center including a fitted Rayleigh distribution as well as a non-parametric kernel density estimate

If package shiny is installed, an interactive web app for this functionality can be run with runGUI("analyze").

Value

A list with the results from the numerical analyses and statistical tests.

sdXY	standard deviations of x- and y-coordinates (in original measurement units, MOA, SMOA, milliradian).
sdXci	parametric (chi^2) and bootstrap confidence intervals for the standard deviation of x-coordinates (in original measurement units, MOA, SMOA, milliradian).
sdYci	parametric (chi^2) and bootstrap confidence intervals for the standard deviation of y-coordinates (in original measurement units, MOA, SMOA, milliradian).
sdXYrob	robust standard deviations of x- and y-coordinates (in original measurement units, MOA, SMOA, milliradian).
covXY	covariance matrix of (x,y)-coordinates.
covXYrob	robust estimate of covariance matrix of (x,y)-coordinates.
distToCtr	mean, median and maximum distance from points to their center as well as estimated Rayleigh parameters sigma (precision), radial standard deviation RSD, and mean radius MR (in original measurement units, MOA, SMOA, milliradian).
sigmaCI	parametric (chi^2) and bootstrap confidence intervals for sigma (in original measurement units, MOA, SMOA, milliradian).
RSDci	parametric (chi^2) and bootstrap confidence intervals for radial standard deviation RSD (number of points), in original measurement units, MOA, SMOA, milliradian).
MRci	parametric (chi^2) and bootstrap confidence intervals for mean radius MR (in original measurement units, MOA, SMOA, milliradian).
maxPairDist	maximum pairwise distance between points (center-to-center, = maximum spread, in original measurement units, MOA, SMOA, milliradian).
groupRect	width and height of bounding box with diagonal and figure of merit FoM (average side length, in original measurement units, MOA, SMOA, milliradian).
groupRectMin	width and height of minimum-area bounding box with diagonal and figure of merit FoM (average side length, in original measurement units, MOA, SMOA, milliradian).
minCircleRad	radius for the minimum enclosing circle (in original measurement units, MOA, SMOA, milliradian).

minEll	length of semi-major and semi-minor axis of the minimum enclosing ellipse (in original measurement units, MOA, SMOA, milliradian).
confEll	length of semi-major and semi-minor axis of the confidence ellipse (in original measurement units, MOA, SMOA, milliradian).
confEllRob	length of semi-major and semi-minor axis of the confidence ellipse based on a robust estimate for the covariance matrix (in original measurement units, MOA, SMOA, milliradian).
confEllShape	aspect ratio of the confidence ellipse (square root of condition index kappa), its flattening (1 - inverse aspect ratio) as well as the trace and determinant of the covariance matrix.
confEllShapeRob	aspect ratio and flattening of the confidence ellipse based on a robust estimate for the covariance matrix as well as its trace and determinant.
CEP	estimate(s) for the circular error probable (CEP, in original measurement units, MOA, SMOA, milliradian).

References

http://ballistipedia.com/index.php?title=Describing_Precision

http://ballistipedia.com/index.php?title=Measuring_Precision

See Also

getDistToCtr, getMaxPairDist, getBoundingBox, getMinBBox, getMinCircle, getConfEll, getCEP, getRayParam, getMOA, hist, boot, boot.ci, kernel, covMcd

Examples

```
# coordinates given by a suitable data frame
res <- groupSpread(DFtalon, CEPtype=c('Grubbs', 'Rayleigh'), CEPlevel=0.5,
                   CIlevel=0.95, bootCI='none', dstTarget=10, conversion='m2mm')
names(res)
res$sdXYrob
res$distToCtr
res$maxPairDist
res$CEP

# coordinates given by a matrix
## Not run:
xy <- matrix(round(rnorm(200, 0, 5), 2), ncol=2)
groupSpread(xy, CEPlevel=0.5, CIlevel=0.95, dstTarget=25, conversion='m2cm')

## End(Not run)
```

Hoyt *The Hoyt Distribution*

Description

Density, distribution function, quantile function, and random deviate generation for the Hoyt distribution. The radius around the true mean in a bivariate normal random variable, re-written in polar coordinates (radius and angle), follows a Hoyt distribution. Equivalently, the modulus of a complex normal random variable does.

Usage

```
dHoyt(x, qpar, omega)
pHoyt(q, qpar, omega, lower.tail = TRUE)
qHoyt(p, qpar, omega, lower.tail = TRUE, loUp = NULL)
rHoyt(n, qpar, omega, method = c('eigen', 'chol', 'cdf'), loUp = NULL)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
qpar	vector of Hoyt shape parameters q. See details.
omega	vector of Hoyt scale parameters omega. See details.
method	string indicating which method to use for generating random deviates. See details.
loUp	search interval for numerical root finding. Either a vector with the lower and upper interval boundary, a list of such vectors, or an (n x 2)-matrix. See details.
lower.tail	logical. If TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.

Details

The parameters qpar and omega may be determined with getHoytParam.

dHoyt involves the modified Bessel function of the first kind and order 0 (Chew & Boye, 1962; Hoyt, 1947). pHoyt is implemented as the symmetric difference between two Marcum Q-functions (Paris, 2009). The Marcum Q-function is the cdf of a non-central chi^2 variable (Nuttall, 1975).

qHoyt is implemented through numerical root finding of pHoyt. If no search interval for uniroot is provided, the quantiles of an approximating central chi^2 distribution are used to determine the search intervals.

rHoyt with method='eigen' or with method='chol' simulates 2D normal deviates based on the covariance matrix corresponding to parameters qpar and omega, and then determines the radius. rHoyt with method='cdf' is much slower as it performs numerical root finding of pHoyt given simulated quantiles from a uniform random variable in (0,1). If no search interval for uniroot

is provided, the quantiles of an approximating central chi^2 distribution are used to determine the search intervals.

See Rice for the distribution of radial error around an offset center for uncorrelated bivariate normal variables with equal variances. See Rayleigh for the distribution of radial error around the true center of uncorrelated bivariate normal variables with equal variances. See mvnEll for the distribution of radial error around an offset center for correlated normal variables with unequal variances.

Value

dHoyt gives the density, pHoyt gives the cumulative distribution function, qHoyt gives the quantile function, rHoyt generates random deviates.

The length of the result is determined by n for rHoyt, and is the maximum of the lengths of the numerical parameters for the other functions.

In dHoyt, pHoyt and qHoyt, the numerical parameters are recycled to the length of the result. Only the first element of the logical parameters is used. In rHoyt, only the first element of qpar and omega is used.

References

Chew, V. & Boyce, R. (1962). Distribution of radial error in bivariate elliptical normal distributions. Technometrics, 4(1), 138-140.

Hoyt, R. S. (1947). Probability functions for the modulus and angle of the normal complex variate. Bell System Technical Journal, 26(2), 318-359.

Nuttall, AH. (1975). Some integrals involving the Q-M function. IEEE Transactions on Information Theory, 21 (1), 95-96

Paris, JF. 2009. Nakagami-q (Hoyt) distribution function with applications. Electronics Letters, 45(4). 210-211. Erratum: doi:10.1049/el.2009.0828

https://reference.wolfram.com/language/ref/HoytDistribution.html

See Also

getHoytParam, Rayleigh, Rice, mvnEll, Bessel, Chisquare, uniroot

Examples

```
dHoyt(1, qpar=c(0.1, 0.5, 0.9), omega=10)
pHoyt(c(0.1, 0.5, 0.9), qpar=0.5, omega=10)
qHoyt(0.5, qpar=0.5, omega=c(5, 10, 15))
rHoyt(5, qpar=0.5, omega=10)
```

`Maxwell` *The Maxwell-Boltzmann Distribution*

Description

Density, distribution function, quantile function, and random deviate generation for the Maxwell-Boltzmann distribution. The radius around the true mean in a trivariate uncorrelated normal random variable with equal variances, re-written in polar coordinates (radius, azimuth, elevation), follows a Maxwell-Boltzmann distribution.

Usage

```
dMaxwell(x, sigma)
pMaxwell(q, sigma, lower.tail = TRUE)
qMaxwell(p, sigma, lower.tail = TRUE)
rMaxwell(n, sigma)
```

Arguments

`x`, `q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`sigma`	vector of parameter sigma (common standard deviation of the underlying normal distribution of each 3D-coordinate).
`lower.tail`	logical. If `TRUE` (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.

Details

The parameter `sigma` may be determined with `getRayParam`.

See `Rayleigh` for the distribution of radial error around the true center of uncorrelated bivariate normal variables with equal variances. See `Hoyt` for the distribution of radial error around the true center of correlated bivariate normal variables with unequal variances. See `Rice` for the distribution of radial error around an offset center for uncorrelated bivariate normal variables with equal variances. See `mvnEll` for the distribution of radial error around an offset center for correlated normal variables with unequal variances.

Value

`dMaxwell` gives the density, `pMaxwell` gives the cumulative distribution function, `qMaxwell` gives the quantile function, `rMaxwell` generates random deviates.

The length of the result is determined by `n` for `rMaxwell`, and is the maximum of the lengths of the numerical parameters for the other functions.

In `dMaxwell`, `pMaxwell` and `qMaxwell` are recycled to the length of the result. Only the first element of the logical parameters is used. In `rRayleigh`, only the first element of `sigma` is used.

References

https://reference.wolfram.com/language/ref/MaxwellDistribution.html

See Also

getRayParam, Rayleigh, Hoyt, Rice, mvnEll

Examples

```
dMaxwell(1, sigma=10)
pMaxwell(c(0.1, 0.5, 0.9), sigma=10)
qMaxwell(0.5, sigma=c(5, 10, 15))
rMaxwell(5, sigma=10)
```

---

mvnEll *Multivariate normal offset ellipse probabilities*

Description

Probability of an offset ellipsoid for a correlated multivariate normal distribution. Offset circle probabilities are a special case.

Usage

```
pmvnEll(r=1, sigma = diag(2), mu, e, x0, lower.tail = TRUE,
        method_cdf = c('integrate', 'saddlepoint'))

qmvnEll(p,   sigma = diag(2), mu, e, x0, lower.tail = TRUE,
        loUp=NULL, method_cdf = c('integrate', 'saddlepoint'))

rmvnEll(n,   sigma = diag(2), mu, e, x0,
        method = c('eigen', 'chol', 'cdf'), loUp=NULL,
        method_cdf = c('integrate', 'saddlepoint'))
```

Arguments

r	vector of radii for the offset ellipse defined by e.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
sigma	true positive definite covariance matrix of multivariate normal distribution.
mu	true center of multivariate normal distribution.
e	positive definite matrix characterizing the offset ellipse defined by (x-x0)' e (x-x0) < r^2. If the ellipse defined by e has semi-axis lengths equal to the square root of the eigenvalues of a matrix S, and is oriented along the eigenvectors of S, then e = S^-1. By default a circle.

x0	center of the offset ellipse.
method	string indicating which method to use for generating random deviates. See details.
loUp	search interval for numerical root finding. Either a vector with the lower and upper interval boundary, a list of such vectors, or an (n x 2)-matrix. See details.
lower.tail	logical. If TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
method_cdf	string indicating which method to use for calculating the sum of non-central chi^2 variables. See details.

Details

pmvnEll is implemented by first transforming the integration region to the unit disc/sphere, then decorrelating the normal distribution through rotation. Finally, the quadratic form (sum of non-central chi^2 variables) is calculated depending on method_cdf. For method_cdf='integrate', numerical integration using farebrother is chosen, for method_cdf='saddlepoint', Kuonen's (1999) saddlepoint approximation. Note that the equation for the cumulant generating function K has a missing zeta in the numerator of the second term in Kuonen (1999), see Imhof (1961) instead. Lower tail probabilities are calculated as '1 - upper tail probability', so loss of accuracy is likely when the upper tail probability is very small (<1e-9).

qmvnEll is implemented through numerical root finding of pmvnEll. If no search interval for uniroot is provided, the quantiles of an approximating non-central chi^2 distribution are used to determine the search intervals.

rmvnEll with method='eigen' or with method='chol' simulates 2D normal deviates based on sigma and mu, and then determines the radius around x0. rmvnEll with method='cdf' is much slower as it performs numerical root finding of pmvnEll given simulated quantiles from a uniform random variable in (0,1). If no search interval for uniroot is provided, the quantiles of an approximating non-central chi^2 distribution are used to determine the search intervals.

See Hoyt for the distribution of radial error around the true center of correlated bivariate normal variables with unequal variances. See Rice for the distribution of radial error around an offset center for uncorrelated bivariate normal variables with equal variances. See Rayleigh for the distribution of radial error around the true center of uncorrelated bivariate normal variables with equal variances.

Value

pmvnEll integrates the multivariate normal distribution over an arbitrary ellipsoid and thus gives the cumulative distribution function. qmvnEll gives the quantile function, rmvnEll generates random deviates.

The functions are vectorized in r and p but not in the remaining parameters.

References

DiDonato, A. R., & Jarnagin, M. P. (1961a). Integration of the general bivariate Gaussian distribution over an offset circle. Mathematics of Computation, 15 (76), 375-382.

DiDonato, A. R., & Jarnagin, M. P. (1961b). Integration of the general bivariate Gaussian distribution over an offset ellipse (NWL TR 1710). Dahlgren, VA: U.S. Naval Weapons Laboratory.

Duchesne, P., & Lafaye de Micheaux, P. (2010). Computing the distribution of quadratic forms: Further comparisons between the Liu-Tang-Zhang approximation and exact methods. Computational Statistics and Data Analysis, 54, 858-862.

Imhof, J. P. (1961). Computing the distribution of quadratic forms in normal variables. Biometrika, 48, 419-426.

Kuonen D. (1999). Saddlepoint Approximations for Distributions of Quadratic Forms in Normal Variables. Biometrika, 86, 929-935.

See Also

Hoyt, farebrother, uniroot

Examples

```
# define a bivariate normal distribution
mu    <- c(2, -1)                       # true mean
sigma <- cbind(c(10, 6), c(6, 10))      # covariance matrix

# define circular integration region
ctr <- c(1, 0)                          # center
e1  <- diag(2)                          # circle
r   <- 2                                # radius
pmvnEll(r,   sigma=sigma, mu=mu, e=e1, x0=ctr) # probability
qmvnEll(0.5, sigma=sigma, mu=mu, e=e1, x0=ctr) # quantile
rmvnEll(5,   sigma=sigma, mu=mu, e=e1, x0=ctr) # random numbers

# define elliptical integration region
S  <- cbind(c(3.5, -0.3), c(-0.3, 1.7))
e2 <- solve(S)
pmvnEll(r,   sigma=sigma, mu=mu, e=e2, x0=ctr) # probability
qmvnEll(0.5, sigma=sigma, mu=mu, e=e2, x0=ctr) # quantile
rmvnEll(5,   sigma=sigma, mu=mu, e=e2, x0=ctr) # random numbers

# plot all regions
evSig <- eigen(sigma)$values
evS   <- eigen(S)$values
xLims <- range(c( mu[1]+c(-1, 1)*sqrt(evSig[1]),
                  ctr[1]+c(-r, r)*sqrt(evS[1])))
yLims <- range(c( mu[2]+c(-1.25, 1.25)*sqrt(evSig[1]),
                  ctr[2]+c(-r, r)*sqrt(evS[1])))

plot(xLims, yLims, type="n", asp=1)
points(mu[1],  mu[2],  pch=16, cex=2, col="black")
points(ctr[1], ctr[2], pch=15, cex=2, col="blue")
drawEllipse(mu, sigma, r=0.75, fg="black")
drawEllipse(mu, sigma, r=1,    fg="black")
drawEllipse(mu, sigma, r=1.25, fg="black")
drawEllipse(mu, sigma, r=1.5,  fg="black")
drawEllipse(ctr, e1, r=r, fg="blue")
drawEllipse(ctr, S,  r=r, fg="red")
legend(x="bottomright", legend=c("normal iso-densities",
```

```
        "integration circle", "integration ellipse"),
        lty=1, col=c("black", "blue", "red"))
```

---

range2CEP *Estimate circular error probable (CEP) based on range statistics*

Description

Estimate the circular error probable (CEP) based on range statistics such as extreme spread, figure of merit, or the bounding box diagonal. This function assumes a circular bivariate normal shot distribution with 0 mean.

Usage

```
range2CEP(x, stat="ES", n=5, nGroups=1, CEPlevel=0.5, CIlevel=0.95,
            collapse=TRUE, dstTarget, conversion)
```

Arguments

x	a numerical vector with values for extreme spread (ES), figure of merit (FoM), or the diagonal of the bounding box (D).
stat	a character vector with elements "ES" (extreme spread), "FoM" (figure of merit), or "D" (bounding box diagonal) indicating which range statistic is given in x. Elements correspond to those in x in the sense that the second element of stat indicates the statistic for the second element of x. If all elements of x are the same kind of statistic, stat only needs to indicate it once.
n	integer between 2 and 100. Number of shots in each group.
nGroups	integer between 1 and 10. Number of groups when x is the average of individually-measured range statistics from several groups.
CEPlevel	a numerical vector with the coverage values for the CEP.
CIlevel	confidence level (coverage probability) for the CEP confidence interval. If one of 0.5, 0.9, 0.95, 0.99, the CI is based on the corresponding quantiles of the Monte Carlo distribution of the range statistic for given n and nGroups. If not, CI can only be calculated for extreme spread using a Patnaik chi^2 approximation to the conditional distribution as suggested by Taylor and Grubbs (1975).
collapse	logical: should the list with CIs be simplified if possible?
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Example 'm2cm'. See getMOA.

Details

Based on the lookup table DFdistr with results form a Monte Carlo simulation. The Rayleigh sigma parameter is estimated using range2sigma, and then converted to CEP with qRayleigh.

If package shiny is installed, an interactive web app for this functionality can be run with runGUI("range").

Value

A list with the calculated CEP values in one component, and the corresponding CIs in the other component.

CEP	The calculated CEP values in the original measurement unit as well as in angular size measures.
CEPCI	The calculated CEP CIs in the original measurement unit as well as in angular size measures.

References

Taylor, M. S., & Grubbs, F. E. (1975). Approximate Probability Distributions for the Extreme Spread (BRL-MR-2438). Aberdeen Proving Ground, MD: U.S. Ballistic Research Laboratory.

See Also

DFdistr, range2sigma qRayleigh getCEP

Examples

```
es  <- getMaxPairDist(DFscar17)$d
fom <- getBoundingBox(DFscar17)$FoM
d   <- getBoundingBox(DFscar17)$diag
range2CEP(c(es, fom, d), stat=c("ES", "FoM", "D"),
          n=nrow(DFscar17), nGroups=1, CEPlevel=0.5, CIlevel=0.9)

# compare with Rayleigh CEP estimate from using
# (x,y)-coordinates of all shots
getCEP(DFscar17, CEPlevel=0.5, type="Rayleigh")
```

---

range2sigma *Estimate Rayleigh sigma based on range statistics*

Description

Estimate the Rayleigh sigma parameter based on range statistics like extreme spread, figure of merit, or the bounding box diagonal. This function assumes a circular bivariate normal shot distribution with 0 mean.

Usage

```
range2sigma(x, stat="ES", n=5, nGroups=1, CIlevel=0.95,
            collapse=TRUE, dstTarget, conversion)
```

Arguments

x	a numerical vector with values for extreme spread (ES), figure of merit (FoM), or the diagonal of the bounding box (D).
stat	a character vector with elements "ES" (extreme spread), "FoM" (figure of merit), or "D" (bounding box diagonal) indicating which range statistic is given in x. Elements correspond to those in x in the sense that the second element of stat indicates the statistic for the second element of x. If all elements of x are the same kind of statistic, stat only needs to indicate it once.
n	integer between 2 and 100. Number of shots in each group.
nGroups	integer between 1 and 10. Number of groups when x is the average of individually-measured range statistics from several groups.
CIlevel	confidence level (coverage probability) for the Rayleigh sigma confidence interval. If one of 0.5, 0.9, 0.95, 0.99, the CI is based on the corresponding quantiles of the Monte Carlo distribution of the range statistic for given n and nGroups. If not, CI can only be calculated for extreme spread using a Patnaik chi^2 approximation to the conditional distribution as suggested by Taylor and Grubbs (1975).
collapse	logical: should the list with CIs be simplified if possible?
dstTarget	a numerical value giving the distance to the target - used in MOA calculation. See getMOA.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Example 'm2cm'. See getMOA.

Details

Based on the lookup table DFdistr with results form a Monte Carlo simulation. If the value of n is not among those simulated (but is less than 100), a monotonic spline interpolation between the neighboring simulated values of the statistic's coefficient of variation is used.

For conversion to the circular error probable, see range2CEP.

Details for the calculation can be found under

http://ballistipedia.com/index.php?title=Range_Statistics

If package shiny is installed, an interactive web app for this functionality can be run with runGUI("range").

Value

A list with the calculated values for sigma in one component, and the corresponding CIs in the other component.

sigma	The calculated values for sigma in the original measurement unit as well as in angular size measures.
sigmaCI	The calculated CIs for sigma in the original measurement unit as well as in angular size measures.

References

Taylor, M. S., & Grubbs, F. E. (1975). Approximate Probability Distributions for the Extreme Spread (BRL-MR-2438). Aberdeen Proving Ground, MD: U.S. Ballistic Research Laboratory.

See Also

DFdistr, range2CEP, efficiency, getRayParam, getMaxPairDist, getBoundingBox

Examples

```
es  <- getMaxPairDist(DFscar17)$d
fom <- getBoundingBox(DFscar17)$FoM
d   <- getBoundingBox(DFscar17)$diag
range2sigma(c(es, fom, d), stat=c("ES", "FoM", "D"),
            n=nrow(DFscar17), nGroups=1, CIlevel=0.9)

# compare with Rayleigh sigma estimate from using
# (x,y)-coordinates of all shots
getRayParam(DFscar17, level=0.9)
```

---

rangeStat *Distribution of range statistics*

Description

Approximate cumulative distribution function, quantile function and random deviates of range statistics based on a lookup table generated by simulations. Includes extreme spread (ES), figure of merit (FoM), bounding box diagonal (D). This function assumes a circular bivariate normal shot distribution with 0 mean.

Usage

```
pRangeStat(q, sigma = 1, nPerGroup = 5, nGroups = 1, stat = c("ES", "FoM", "D"),
           lower.tail = TRUE, loUp)
qRangeStat(p, sigma = 1, nPerGroup = 5, nGroups = 1, stat = c("ES", "FoM", "D"),
           method = c("linear", "spline"), lower.tail = TRUE)
rRangeStat(n, sigma = 1, nPerGroup = 5, nGroups = 1, stat = c("ES", "FoM", "D"))
```

Arguments

q	vector of quantiles.
p	vector of probabilities. Must be within [0.005, 0.995].
n	number of observations. Must be <= 100. If `length(n) > 1`, the length is taken to be the number required.
stat	character string indicating the range statistic. One of "ES" (extreme spread), "FoM" (figure of merit), or "D" (bounding box diagonal).
nPerGroup	integer between 2 and 100. Number of shots in each group.

nGroups	integer between 1 and 10. Number of groups. For nGroups > 1, the quantile of the average range statistic is returned.
sigma	numeric value > 0 indicating the Rayleigh scale parameter (common standard deviation of the underlying normal distribution of each 2D-coordinate). See Rayleigh.
lower.tail	logical. If TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
method	method for bivariate interpolation if both, p and nPerGroup, are missing from the lookup table. "linear" for bilinear interpolation (requires installing package interp), "spline" for bivariate spline approximation (requires installing package MBA).
loUp	search interval for numerical root finding. A vector with the lower and upper interval boundary. See details.

Details

Based on the lookup table DFdistr with results form a Monte Carlo simulation. If the value either for p or for nPerGroup is missing from the lookup table, a monotone spline interpolation between the neighboring simulated values is used.

pRangeStat is implemented through numerical root finding of qRangeStat. If no search interval for uniroot is provided, whole interval of probabilities available in DFdistr is used. NA is returned for quantiles corresponding to probabilities outside of the available range.

Value

pRangeStat gives the cumulative distribution function, qRangeStat gives the quantile function, rRangeStat generates random deviates.

References

http://ballistipedia.com/index.php?title=Range_Statistics

See Also

DFdistr, range2sigma, Rayleigh, interp, mba.surf,

Examples

```
(q45 <- pRangeStat(c(4, 5), sigma=1.5, n=5, nGroups=3, stat="ES"))

# should be the 4 and 5
qRangeStat(q45, sigma=1.5, n=5, nGroups=3, stat="ES")

rRangeStat(5, sigma=2, nPerGroup=5, nGroups=3, stat="D")
```

Rayleigh *The Rayleigh Distribution*

Description

Density, distribution function, quantile function, and random deviate generation for the Rayleigh distribution. The radius around the true mean in a bivariate uncorrelated normal random variable with equal variances, re-written in polar coordinates (radius and angle), follows a Rayleigh distribution.

Usage

```
dRayleigh(x, scale)
pRayleigh(q, scale, lower.tail = TRUE)
qRayleigh(p, scale, lower.tail = TRUE)
rRayleigh(n, scale)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
scale	vector of Rayleigh scale parameters (common standard deviation of the underlying normal distribution of each 2D-coordinate).
lower.tail	logical. If TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.

Details

The parameter scale may be determined with getRayParam.

See Maxwell for the distribution of radial error around the true center of uncorrelated trivariate normal variables with equal variances. See Hoyt for the distribution of radial error around the true center of correlated bivariate normal variables with unequal variances. See Rice for the distribution of radial error around an offset center for uncorrelated bivariate normal variables with equal variances. See mvnEll for the distribution of radial error around an offset center for correlated normal variables with unequal variances.

Value

dRayleigh gives the density, pRayleigh gives the cumulative distribution function, qRayleigh gives the quantile function, rRayleigh generates random deviates.

The length of the result is determined by n for rRayleigh, and is the maximum of the lengths of the numerical parameters for the other functions.

In dRayleigh, pRayleigh and qRayleigh, the numerical parameters are recycled to the length of the result. Only the first element of the logical parameters is used. In rRayleigh, only the first element of scale is used.

References

https://reference.wolfram.com/language/ref/RayleighDistribution.html

See Also

getRayParam, Maxwell, Rice, Hoyt, mvnEll

Examples

```
dRayleigh(1, scale=10)
pRayleigh(c(0.1, 0.5, 0.9), scale=10)
qRayleigh(0.5, scale=c(5, 10, 15))
rRayleigh(5, scale=10)
```

---

readDataMisc *Read data from text files*

Description

Reads data from text files that have a similar structure to OnTarget PC/TDS output files, specifically from Taran. Several files can be read with one call.

Usage

```
readDataMisc(fPath = ".", fNames, fPat, combine = TRUE,
             dstTarget, conversion)
```

Arguments

fPath	a character string containing the path to the folder with the data files, e.g. 'c:/folder/otFiles'.
fNames	a character vector containing the file names of the files that should be read in.
fPat	a character string containing the regular-expression that describes all names of files that should be read in. E.g., '^points[[:digit:]]{2}\\.txt$' for file-names 'points**.txt', where ** are 2 digits. See regex, glob2rx.
combine	logical: combine the data into one big data frame with combineData?
dstTarget	a numerical value/vector giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in the data.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Example 'm2cm'. See getMOA.

Details

If `fNames` is provided, `fPat` is ignored.

If neither `fNames` nor `fPat` is provided, and we are in interactive mode under Windows, files can be chosen interactively.

This function is basically a wrapper for `read.table` and `read.csv`.

The files need to be either comma-separated or whitespace-delimited, and have a header with the variable names. Variable names must not contain spaces. In order to be later used by functions `analyzeGroup` or `compareGroups`, coordinates for points of impact must be given with `point.x`, `point.y`, `x`, `y`, or `ShotX`, `ShotY`. Point of aim can be given with `aim.x`, `aim.y`, otherwise (0,0) will be assumed in analysis functions. Distance to target can be given with `distance`, otherwise 100m will be assumed in analysis functions. Files should to contain the variable `Group` if `combineData` should be later used to combine them into one big data frame. There can only be exactly as many variable names as there are non-empty data-columns.

For reading in files exported from OnTarget PC 1.*, see `readDataOT1`.
For reading in files exported from OnTarget PC 2.* or OnTarget TDS v3.*, see `readDataOT2`.

Value

With `combine=FALSE`: a list of data frames, each from one file that was read in. This list can then be combined into one big data frame by `combineData`.

`file1`	data frame containing data from the first file
`file2`	data frame containing data from the second file
`...`	more data frames

With `combine=TRUE`: a combined data frame from `combineData`.

See Also

`read.table`, `regex`, `glob2rx`, `combineData`, `readDataOT1`, `readDataOT2`, `readDataSMT`, `readDataShotMarker`

Examples

```
## Not run:
fPath   <- 'c:/folder/files'             # folder with data files
fileNam <- c('pts01.txt', 'pts02.txt')   # desired files in that folder
DFgroup <- readDataMisc(fPath, fNames=fileNam, combine=TRUE)

## alternatively, specify filename pattern for all files to be read in
fPath   <- 'c:/folder/otFiles'           # folder with data files
fPat    <- '^pts[[:digit:]]{2}\\.txt$' # filename pattern
DFgroup <- readDataMisc(fPath, fPat=pattern, combine=TRUE)

## End(Not run)

## result should look like this
data(DFcm)
head(DFcm)
```

readDataOT1 *Read data files exported by OnTarget PC v1.1**

Description

Reads in data from files exported by OnTarget PC v1.1*. Several files can be read with one call.

Usage

```
readDataOT1(fPath = ".", fNames, fPat, combine = TRUE,
            dstTarget, conversion)
```

Arguments

fPath	a character string containing the path to the folder with the OnTarget PC output files. E.g., 'c:/folder/otFiles'.
fNames	a character vector containing the file names of the files that should be read in.
fPat	a character string containing the regular-expression that describes all names of files that should be read in. E.g., '^points[[:digit:]]{2}\\.txt$' for filenames 'points**.txt', where ** are 2 digits. See regex, glob2rx.
combine	logical: combine the data into one big data frame with combineData?
dstTarget	a numerical value/vector giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in the data.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Example 'm2cm'. See getMOA.

Details

If fNames is provided, fPat is ignored.

If neither fNames nor fPat is provided, and we are in interactive mode under Windows, files can be chosen interactively.

This function is basically a wrapper for read.delim.

Output files need to be tab-delimited files from OnTarget PC v1.1*: 'Tools -> Export Point Data'. Files need a trailing tab in each row, and need to contain exactly the following variable names in this order: Project Title, Group, Ammunition, Distance, Aim X, Aim Y, Center X, Center Y, Point X, Point Y.

Tested with OnTarget PC v1.10.
For reading in files exported from OnTarget PC v2.* or OnTarget TDS v3.*, see readDataOT2.
For reading in other text files, see readDataMisc.

Value

With `combine=FALSE`: a list of data frames, each from one file that was read in. This list can then be combined into one big data frame by `combineData`.

`file1`	data frame containing data from the first file
`file2`	data frame containing data from the second file
`...`	more data frames

With `combine=TRUE`: a combined data frame from `combineData`.

See Also

`read.delim`, `regex`, `glob2rx`, `combineData`, `readDataOT2`, `readDataSMT`, `readDataShotMarker`, `readDataMisc`

Examples

```
## Not run:
# folder with OnTarget PC v1.1* output files
fPath   <- 'c:/folder/otFiles'
fileNam <- c('pts01.txt', 'pts02.txt')   # desired files in that folder
DFgroup <- readDataOT1(fPath, fNames=fileNam)

## alternatively, specify filename pattern for all files to be read in
fPath   <- 'c:/folder/otFiles'           # folder with data files
fPat    <- '^pts[[:digit:]]{2}\\.txt$' # filename pattern
DFgroup <- readDataOT1(fPath, fPat=pattern)

## End(Not run)

## result should look like this
data(DFcm)
head(DFcm)
```

`readDataOT2` *Read data files exported by OnTarget PC v2.* or OnTarget TDS v3.**

Description

Reads in data from files exported by OnTarget PC v2.* or OnTarget TDS v3.*. Several files can be read with one call.

Usage

```
readDataOT2(fPath = ".", fNames, fPat, combine = TRUE,
            dstTarget, conversion)
```

Arguments

fPath	a character string containing the path to the folder with the OnTarget PC/TDS output files, e.g. 'c:/folder/otFiles'.
fNames	a character vector containing the file names of the files that should be read in.
fPat	a character string containing the regular-expression that describes all names of files that should be read in. E.g., '^points[[:digit:]]{2}\\.txt$' for filenames 'points**.txt', where ** are 2 digits. See regex, glob2rx.
combine	logical: combine the data into one big data frame with combineData?
dstTarget	a numerical value/vector giving the distance to the target - used in MOA calculation. Acts as override if variable distance is already included in the data.
conversion	how to convert the measurement unit for distance to target to that of the (x,y)-coordinates in MOA calculation. Example 'm2cm'. See getMOA.

Details

If fNames is provided, fPat is ignored.

If neither fNames nor fPat is provided, and we are in interactive mode under Windows, files can be chosen interactively.

This function is basically a wrapper for read.csv.

Output files need to be comma-separated files (file type .csv) from OnTarget PC v2.* or OnTarget TDS v3.*: 'Tools -> Export Point Data'. Files need to contain exactly the following variable names in this order: Project Title, Group, Ammunition, Distance, Aim X, Aim Y, Center X, Center Y, Point X, Point Y, and optionally Velocity.

Tested with OnTarget PC v2.10 and v2.28 as well as OnTarget TDS v3.71, v3.89, v6.09.
For reading in files exported from OnTarget PC v1.*, see readDataOT1.
For reading in other text files, see readDataMisc.

Value

With combine=FALSE: a list of data frames, each from one file that was read in. This list can then be combined into one big data frame by combineData.

file1	data frame containing data from the first file
file2	data frame containing data from the second file
...	more data frames

With combine=TRUE: a combined data frame from combineData.

See Also

read.csv, regex, glob2rx, combineData, readDataOT1, readDataSMT, readDataShotMarker, readDataMisc

Examples

```
## Not run:
# folder with OnTarget PC v2.* or OnTarget TDS v3.* output files
fPath   <- 'c:/folder/otFiles'
fileNam <- c('pts01.csv', 'pts02.csv')   # desired files in that folder
DFgroup <- readDataOT2(fPath, fNames=fileNam, combine=TRUE)

## alternatively, specify filename pattern for all files to be read in
fPath   <- 'c:/folder/otFiles'            # folder with data files
fPat    <- '^pts[[:digit:]]{2}\\.txt$'  # filename pattern
DFgroup <- readDataOT2(fPath, fPat=pattern, combine=TRUE)

## End(Not run)

## result should look like this
data(DFcm)
head(DFcm)
```

---

readDataShotMarker *Read data files exported by the ShotMarker e-target system*

Description

Reads in data from files exported by the ShotMarker e-target system. Either CSV files or backup files. Several files can be read with one call.

Usage

```
readDataShotMarker(fPath = ".", fNames, fPat, combine = TRUE)
```

Arguments

fPath	a character string containing the path to the folder with the ShotMarker output files, e.g. 'c:/folder/smFiles'.
fNames	a character vector containing the file names of the files that should be read in. Can be either plain text CSV file(s) or complete .tar backup file(s).
fPat	a character string containing the regular-expression that describes all names of files that should be read in. E.g., '^points[[:digit:]]{2}\\.txt$' for filenames 'points**.txt', where ** are 2 digits. See regex, glob2rx.
combine	logical: combine the data into one big data frame with combineData?

Details

If fNames is provided, fPat is ignored.

If neither fNames nor fPat is provided, and we are in interactive mode under Windows, files can be chosen interactively.

For CSV files, this function is basically a wrapper for read.csv. To read in backup files, package jsonlite must be installed.

Output files can be comma-separated files (file type .csv) from ShotMarker e-target.

Coordinates are stored in inch, distance is converted to yard.

Value

With combine=FALSE: a list of data frames, each from one file that was read in. This list can then be combined into one big data frame by combineData.

file1	data frame containing data from the first file
file2	data frame containing data from the second file
...	more data frames

With combine=TRUE: a combined data frame from combineData.

See Also

read.csv, regex, glob2rx, combineData, readDataOT1, readDataOT2, readDataSMT, readDataMisc

Examples

```
## Not run:
# folder with Silver Mountain e-target output files
fPath   <- 'c:/folder/smtFiles'
fileNam <- c('pts01.csv', 'pts02.csv')    # desired files in that folder
DFgroup <- readDataShotMarker(fPath, fNames=fileNam, combine=TRUE)

## alternatively, specify filename pattern for all files to be read in
fPath   <- 'c:/folder/smtFiles'           # folder with data files
fPat    <- '^pts[[:digit:]]{2}\\.csv$'    # filename pattern
DFgroup <- readDataShotMarker(fPath, fPat=pattern, combine=TRUE)

## End(Not run)

## result should look like this
data(DFcm)
head(DFcm)
```

---

readDataSMT *Read data files exported by the Silver Mountain e-target system*

Description

Reads in data from CSV files exported by the Silver Mountain e-target system. Several files can be read with one call.

Usage

```
readDataSMT(fPath = ".", fNames, fPat, combine = TRUE)
```

Arguments

fPath	a character string containing the path to the folder with the SMT output files, e.g. 'c:/folder/smtFiles'.
fNames	a character vector containing the file names of the files that should be read in.
fPat	a character string containing the regular-expression that describes all names of files that should be read in. E.g., '^points[[:digit:]]{2}\\.txt$' for filenames 'points**.txt', where ** are 2 digits. See regex, glob2rx.
combine	logical: combine the data into one big data frame with combineData?

Details

If fNames is provided, fPat is ignored.

If neither fNames nor fPat is provided, and we are in interactive mode under Windows, files can be chosen interactively.

This function is basically a wrapper for read.csv.

Output files need to be comma-separated files (file type .csv) from Silver Mountain e-target. Files need to contain exactly the following variable names in this order: string, shooter, frame, distance, date, score, moa_x, moa_y, scope_x, scope_y, adj_x, adj_y, v, adj_y_avg, adj_y_sd, v_avg, v_sd.

Coordinates are converted from MOA to inch, distance is converted from meter to yard.

For reading in other text files, see readDataMisc.

Value

With combine=FALSE: a list of data frames, each from one file that was read in. This list can then be combined into one big data frame by combineData.

file1	data frame containing data from the first file
file2	data frame containing data from the second file
...	more data frames

With combine=TRUE: a combined data frame from combineData.

See Also

read.csv, regex, glob2rx, combineData, readDataOT1, readDataOT2, readDataShotMarker, readDataMisc

Examples

```
## Not run:
# folder with Silver Mountain e-target output files
fPath   <- 'c:/folder/smtFiles'
fileNam <- c('pts01.csv', 'pts02.csv')    # desired files in that folder
DFgroup <- readDataSMT(fPath, fNames=fileNam, combine=TRUE)

## alternatively, specify filename pattern for all files to be read in
fPath   <- 'c:/folder/smtFiles'           # folder with data files
fPat    <- '^pts[[:digit:]]{2}\\.csv$'    # filename pattern
DFgroup <- readDataSMT(fPath, fPat=pattern, combine=TRUE)

## End(Not run)

## result should look like this
data(DFcm)
head(DFcm)
```

Rice *The Rice Distribution*

Description

Density, distribution function, quantile function, and random deviate generation for the Rice distribution. The radius around the origin in a bivariate uncorrelated normal random variable with equal variances and an offset mean, re-written in polar coordinates (radius and angle), follows a Rice distribution.

Usage

```
dRice(x, nu, sigma)
pRice(q, nu, sigma, lower.tail = TRUE)
qRice(p, nu, sigma, lower.tail = TRUE)
rRice(n, nu, sigma, method = c('eigen', 'chol', 'cdf'))
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
nu	vector of Rice shape parameters nu. See details.
sigma	vector of Rice scale parameter sigma. See details.
method	string indicating which method to use for generating random deviates. See details.
lower.tail	logical. If TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.

Details

dRice involves the modified Bessel function of the first kind and order 0. pRice and qRice are implemented using the Marcum Q-function. The Marcum Q-function is the cdf of a non-central chi^2 variable (Nuttall, 1975).

rRice with method='eigen' or with method='chol' simulates 2D normal deviates based on the diagonal covariance matrix with entries sigma^2, and then determines the radius. rRice with method='cdf' uses qRice with simulated quantiles from a uniform random variable in (0,1).

See Hoyt for the distribution of radial error around the true center of correlated bivariate normal variables with unequal variances. See Rayleigh for the distribution of radial error around the true center of uncorrelated bivariate normal variables with equal variances. See mvnEll for the distribution of radial error around an offset center for correlated normal variables with unequal variances.

For very large signal-to-noise ratios (nu/sigma), a normal approximation is used.

Value

dRice gives the density, pRice gives the cumulative distribution function, qRice gives the quantile function, rRice generates random deviates.

The length of the result is determined by n for rRice, and is the maximum of the lengths of the numerical parameters for the other functions.

In dRice, pRice and qRice, the numerical parameters are recycled to the length of the result. Only the first element of the logical parameters is used. In rRice, only the first element of nu and sigma is used.

References

Nuttall, AH. (1975). Some integrals involving the Q-M function. IEEE Transactions on Information Theory, 21 (1), 95-96

https://reference.wolfram.com/language/ref/RiceDistribution.html

See Also

Rayleigh, Hoyt, mvnEll, Bessel, Chisquare, uniroot

Examples

```
dRice(1, nu=c(0.1, 0.5, 0.9), sigma=10)
pRice(c(0.1, 0.5, 0.9), nu=0.5, sigma=10)
qRice(0.5, nu=0.5, sigma=c(5, 10, 15))
rRice(5, nu=0.5, sigma=10)
```

runGUI *Open web-based GUI in browser*

Description

Opens one of four web-based GUIs for shotGroups functionality in an external browser.

Usage

```
runGUI(app=c("analyze", "hitprob", "range", "angular"), ...)
```

Arguments

app	character string. One of "analyze" - shot group analysis based on data with (x,y)-coordinates of shots, "hitprob" - calculate hit probability within a region, "range" - use measured range statistics to estimate Rayleigh sigma parameter or get required number of groups to achieve a desired CI width, "angular" - angular size conversions
...	arguments passed to runApp. Supply port=80 if a web browser refuses to connect to the randomly chosen port for security reasons. Requires installing packageshiny first.

Details

Requires installing package shiny first. This function calls runApp to run the included shotGroupsApp application.

See Also

runApp

Examples

```
## Not run:
runGUI(app="analzye")

## End(Not run)
```

`simRingCount` *Calculate simulated ring count for a given group and target*

Description

Calculates the simulated ring count given a group, bullet diameter, and target type.

Usage

```
simRingCount(xy, center = FALSE, target, caliber, unit = 'cm')

## S3 method for class 'data.frame'
simRingCount(xy, center = FALSE, target, caliber, unit = 'cm')

## Default S3 method:
simRingCount(xy, center = FALSE, target, caliber, unit='cm')
```

Arguments

`xy`	either a numerical (n x 2)-matrix with the (x,y)-coordinates of n points (1 row of coordinates per point), or a data frame with either the variables `x`, `y` or `point.x`, `point.y` as well as `aim.x`, `aim.y` giving the point of aim (= bullseye). If missing, point of aim (bullseye) is assumed to be in (0,0).
`center`	logical: center groups to mean (0,0) first? If variable `series` does not exist, all shots are treated as belonging to the same group. Only available in method `simRingCount.data.frame()`.
`target`	either a character value with the name of a target in `targets` or a list with a target definition containing the same components as those in `targets` (`name`, `unitTarget`, `nRings`, `ringD10`, `ringD10i`, `ringW`, `cols`, `colsTxt`).
`caliber`	a numerical value indicating the bullet diameter in mm.
`unit`	measurement unit of the (x,y)-coordinates in `xy`. Possible values are `'cm'`, `'mm'`, `'m'`, `'in'`, `'ft'`, `'yd'`.

Details

The returned ring count assumes that bullet holes exactly have the diameter given by `caliber`, and that rings exactly have the diameter/width given in the definition of `target`. The count thus ignores the possibility of ragged bullet holes as well as the physical width of the ring markings. The simulated ring count therefore need not be equal to the calculated ring count from the corresponding physical target.

Value

A list with the following components:

`count`	the total ring count.
`max`	the maximum ring count achievable with the given number of shots.
`rings`	the individual ring count for each shot.

See Also

targets, getDistToCtr

Examples

```
simRingCount(DFscar17, target='ISSF_100m', caliber=5.56, unit='in')

# ring count for all groups in DFcm data set
rc <- by(DFcm, DFcm$series, FUN=simRingCount, target='BDS9',
         caliber=9, unit='cm')

sapply(rc, function(x) with(x, c(count=count, max=max)))
```

---

targets *List containing definitions of several circular target types from the shooting federations ISSF, NRA, DSB, BDS, BDMP, DSU*

Description

List containing definitions of several circular german (DSB, BDS, BDMP, DSU), ISSF, and NRA target types.

Usage

```
data(targets)
```

Format

A list with the following components, each defining one target type.

`ISSF_10mAR` ISSF 10m Air Rifle.

`ISSF_10mAP` ISSF 10m Air Pistol.

`ISSF_25mPP` ISSF 25m Precision Pistol, 50m Pistol.

`ISSF_25mRFP` ISSF 25m Rapid Fire Pistol.

`ISSF_50m` ISSF 50m Rifle.

`ISSF_100m` ISSF 100m Rifle (same as `ISSF_25mPP`).

`ISSF_300m` ISSF 300m Rifle.

`ISSF_25ydPP` ISSF 25m and 50m Precision Pistol target adapted to 25yd.

`ISSF_50ydPP` ISSF 25m and 50m Precision Pistol target adapted to 50yd.

`ISSF_50ftPP` ISSF 25m and 50m Precision Pistol target adapted to 50ft.

`ISSF_50ftSP` ISSF 25m and 50m Precision Pistol target adapted to 50ft Sport Pistol.

`ISSF_25ydRFP` ISSF 25m Rapid Fire Pistol target adapted to 25yd.

`ISSF_50ftRFP` ISSF 25m Rapid Fire Pistol target adapted to 50ft.

`ISSF_50ft` ISSF 50m Rifle target adapted to 50ft.

ISSF_50yd ISSF 50m Rifle target adapted to 50yd.

ISSF_100yd ISSF 300m Rifle target adapted to 100yd.

ISSF_200yd ISSF 300m Rifle target adapted to 200yd.

ISSF_300yd ISSF 300m Rifle target adapted to 300yd.

NRA_HPR_SR NRA 200yd High Powered Rifle SR: Military Target, Rifle Competition, Short Range.

NRA_HPR_SR3 NRA 300yd High Powered Rifle SR-3.

NRA_P_B16 NRA 25yd Pistol Slow Fire B-16.

NRA_MR-1 NRA MR-1.

NRA_MR-1FC NRA MR-1 F-class.

NRA_MR-63 NRA MR-63 300yd.

NRA_MR-63FC NRA MR-63 F-class.

NRA_MR-65 NRA MR-65 500yd.

NRA_MR-65FC NRA MR-65 F-class.

NRA_LR NRA LR.

NRA_LRFC NRA LR F-class.

DSB1 DSB 10m Luftgewehr (same as ISSF_10mAR).

DSB2 DSB 15m Zimmerstutzen.

DSB3 DSB 50m Kleinkalibergewehr (same as ISSF_50m).

DSB4 DSB 100m Kleinkalibergewehr, 25m Pistole-Praezision, 25m Standardpistole, 50m Pistole (same as ISSF_25mPP).

DSB5 DSB 300m Gewehr/Vorderlader Freigewehr (same as ISSF_300m).

DSB6 DSB 50m Muskete Luntenschlossgewehr.

DSB7 DSB 10m Luftpistole (same as ISSF_10mAP).

DSB9 DSB 25m Schnellfeuerpistole, 25m Pistole Duell (same as ISSF_25mRFP).

DSB11 DSB 10m Laufende Scheibe.

BDS1 BDS 100m.

BDS2 BDS 50m Zielfernrohr.

BDS3 BDS 50m (same as ISSF_50m).

BDS4 BDS 100m Zielfernrohr.

BDS5 BDS Pistole 25m, Pistole 50m (same as ISSF_100m).

BDS7 BDS 300m (same as ISSF_300m).

BDS8 BDS 300m Zielfernrohr.

BDS9 BDS 25m Kurzwaffe.

BDS13 BDS 10m Luftgewehr (same as ISSF_10mAR).

BDS14 BDS 10m Luftpistole (same as DSB7).

BDMP1_25m BDMP 25m .30 M1 Carbine.

BDMP1_50m BDMP 50m .30 M1 Carbine.

`BDMP1_100m` BDMP 100m SG 1, CISM-Gewehr, .30 M1 Carbine.

`BDMP2` BDMP 300m SG 2, CISM-Gewehr, DG 2, FG 1, PVDG 1, PHDG 1, SDG 1, SDG 2, PFG 1, SPPDG 1 (same as `ISSF_300m`).

`BDMP3` BDMP ZG 1.

`BDMP4` BDMP ZG 2, ZG 3, ZG 4.

`BDMP5` BDMP 300m DG 3, DG 4.

`DSUa2` DSU a2.

`DSUb2` DSU b2.

`DSUb3` DSU b3.

`DSUb4` DSU b4.

`DSUb5` DSU b5.

`DSUb5P` DSU b5 Praezision.

`DSU_200mP` DSU 200m Praezision.

`DSU_UITP` DSU UIT Praezision (same as `ISSF_100m`).

`DSU_KKI` DSU KK international (same as `ISSF_50m`).

Details

Each target is defined by (at least) the following parameters

- `name`: target name
- `unitTarget`: measurement unit for ring diameters and radii
- `nRings`: number of rings
- `maxCount`: highest ring count for scoring
- `ringD10`: diameter of ring number 10 (highest-valued ring)
- `ringD10i`: diameter of sub-division of ring number 10 (Innenzehn). If target has no sub-division, equal to `ringD10`
- `ringW`: width of the remaining rings number 9, 8, 7, ...
- `cols`: `nRings`+1 colors of the rings - right half of the target, starting with the sub-division of ring number 10 and going outwards
- `colsTxt`: `nRings`-1 colors of the ring numbers, starting with ring number 9 and going outwards

A target may have more parameters, e.g., `draw` to indicate the drawing function that should be used, or `countMouche` if the inner 10 (mouche) should be counted extra in scoring.

See Also

`drawTarget`, `drawGroup`

Examples

```
data(targets)
names(targets)

targets$ISSF_25mPP
```

Index