Package: bulletr (via r-universe)

October 28, 2024

Title Algorithms for Matching Bullet Lands

Version 0.1.0.9003

License MIT

Date 2018-02-08

Description Analyze bullet lands using nonparametric methods. We provide a reading routine for x3p files (see <http://www.openfmc.org> for more information) and a host of analysis functions designed to assess the probability that two bullets were fired from the same gun barrel.

URL https://github.com/csafe-isu/bulletr

BugReports https://github.com/csafe-isu/bulletr/issues

Imports xml2, zoo, ggplot2, plyr, dplyr, reshape2, plotly, robustbase, smoother, readr (>= 1.1.0), digest (>= 0.6.12), rgl, x3ptools

Depends R (>= 3.1)

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 6.0.1

LazyData true

Repository https://erichare.r-universe.dev

RemoteUrl https://github.com/erichare/bulletr

RemoteRef HEAD

RemoteSha a10d16a6573250e68bcd704eca3e101cc5354279

Contents

`boot_fit_loess` *Fit a LOESS model with bootstrap samples*

Description

Fit a LOESS model with bootstrap samples

Usage

```
boot_fit_loess(bullet, groove, B = 1000, alpha = 0.95)
```

Arguments

bullet	Bullet as returned from x3p_to_df
groove	Groove as returned from get_grooves
B	number of Bootstrap samples
alpha	The significance level

---

br411 *3d topological surface measurements for one land of a bullet from the Hamby study*

Description

Some more info - not sure at the moment which bullet this is. Describe structure.

Usage

```
br411
```

Format

a list

---

bulletAlign *Align two surface cross cuts according to maximal correlation*

Description

The bullet with the first name serves as a reference, the second bullet is shifted.

Usage

```
bulletAlign(data, value = "l30")
```

Arguments

data	data frame consisting of at least two surface crosscuts as given by function bulletSmooth.
value	string of the variable to match. Defaults to l30, the variable returned from function bulletSmooth.

Value

list consisting of a) the maximal cross correlation, b) the lag resulting in the highest cross correlation, and c) same data frame as input, but y vectors are aligned for maximal correlation

---

`bulletCheckCrossCut` *Identifying a reliable cross section*

Description

Identifies a "representative" cross section for a bullet land engraved area. Striation marks on a bullet land are the best expressed at the heel (bottom) of a bullet where break-off is still problematic. Using cross-correlation we identify a cross section that is the closest to the bottom of the bullet but does not suffer from break-off. If the resulting cross section is equal to the maximum of the search area (defined in xlimits), there should be some investigation, whether this cross section is usable. There is the risk of tank rash. XXX still to do: are missing values only on the right hand side (leading shoulder)?

Usage

```
bulletCheckCrossCut(path, bullet = NULL, distance = 25, xlimits = c(50,
  500), minccf = 0.9, span = 0.03, percent_missing = 50)
```

Arguments

`path`	path to an x3p file. Ignored in case bullet is not NULL.
`bullet`	if not NULL, the bullet land engraved area (in x3p format).
`distance`	positive numeric value indicating the distance between cross sections to use for a comparison
`xlimits`	vector of values between which to check for cross sections in a stable region
`minccf`	minimal value of cross correlation to indicate a stable region
`span`	The span for the loess smooth function
`percent_missing`	maximum percent missing values on the crosscut to be picked

bulletGetMaxCMS *Identify the number of maximum CMS between two bullet lands*

Description

Identify the number of maximum CMS between two bullet lands

Usage

```
bulletGetMaxCMS(lof1, lof2, column = "resid", span = 35)
```

Arguments

lof1	dataframe of smoothed first signature
lof2	dataframe of smoothed second signature
column	The column which to smooth
span	positive number for the smoothfactor to use for assessing peaks.

Value

list of matching parameters, data set of the identified striae, and the aligned data sets.

bulletGetMaxCMS_nist *Identify the number of maximum CMS between two bullet lands*

Description

Identify the number of maximum CMS between two bullet lands

Usage

```
bulletGetMaxCMS_nist(lof1, lof2, column = "resid", span = 35)
```

Arguments

lof1	dataframe of smoothed first signature
lof2	dataframe of smoothed second signature
column	The column which to smooth
span	positive number for the smoothfactor to use for assessing peaks.

Value

list of matching parameters, data set of the identified striae, and the aligned data sets.

bulletSmooth *Smooth the surface of a bullet*

Description

Smooth the surface of a bullet

Usage

```
bulletSmooth(data, span = 0.03, limits = c(-5, 5), id = "bullet")
```

Arguments

data	data frame as returned by the function processBullets
span	width of the smoother, defaults to 0.03
limits	vector of the form c(min, max). Results will be limited to be between these values.
id	variable name of the land identifier

Value

data frame of the same form as the input extended by the vector l30 for the smooth.

chumbley *Chumbley test score*

Description

Chumbley test score

Usage

```
chumbley(b1, b2, window, reps = 3)
```

Arguments

b1	dataframe
b2	dataframe
window	width of the window (in indices) to consider for matching
reps	number of replicates to use in the evaluation

Examples

```
library(dplyr)

data(br411)
b1 <- get_crosscut(x = 250, bullet=br411)
b2 <- get_crosscut(x = 150, bullet = br411)
b3 <- get_crosscut(x = 10, bullet=br411)
b1.gr <- b1 %>% get_grooves(smoothfactor=30)
b2.gr <- b2 %>% get_grooves()
b3.gr <- b3 %>% get_grooves()
# get signatures
b1 <- fit_loess(b1, b1.gr)$data
b2 <- fit_loess(b2, b2.gr)$data
b3 <- fit_loess(b3, b3.gr)$data
match12 <- get_lag_max_R(b1, b2, window = 100, b1.left = 450)
# matched correlations
get_cor(b1, b2, window = 100, b1.left = 800, lag = match12$lag)
get_cor(b1, b2, window = 100, b1.left = 1000, lag = match12$lag)
get_cor(b1, b2, window = 100, b1.left = 1200, lag = match12$lag)
# random correlations
get_cor(b1, b2, window = 100, b1.left = 800, lag = 100)
get_cor(b1, b2, window = 100, b1.left = 1000, lag = -300)
get_cor(b1, b2, window = 100, b1.left = 1200, lag = -500)

chumbley(b1, b2, window=150, reps=5)

match13 <- get_lag_max_R(b1, b3, window = 100, b1.left = 450)
# matched correlations
get_cor(b1, b3, window = 100, b1.left = 800, lag = match13$lag)
get_cor(b1, b3, window = 100, b1.left = 1000, lag = match13$lag)
get_cor(b1, b3, window = 100, b1.left = 1200, lag = match13$lag)
# random correlations
get_cor(b1, b3, window = 100, b1.left = 800, lag = 100)
get_cor(b1, b3, window = 100, b1.left = 1000, lag = 300)
get_cor(b1, b3, window = 100, b1.left = 1200, lag = 500)

chumbley(b1, b3, window=100, reps=5)
```

CMS *Table of the number of consecutive matches*

Description

Table of the number of consecutive matches

Usage

```
CMS(match)
```

Arguments

match	is a Boolean vector of matches/non-matches

Value

a table of the number of the CMS and their frequencies

Examples

```
x <- rbinom(100, size = 1, prob = 1/3)
CMS(x == 1) # expected value for longest match is 3
```

---

compute_average_scores

Get average scores

Description

Get average scores

Usage

```
compute_average_scores(land1, land2, score)
```

Arguments

land1	(numeric) vector with land ids of bullet 1
land2	(numeric) vector with land ids of bullet 2
score	numeric vector of scores to be summarized

Value

numeric vector of average scores. Length is the same as the number of land engraved areas on the bullets.

do_align *Align two surface cross cuts using cross correlation*

Description

The first vector serves as a reference, the second vector is shifted, such that it aligns best with the first and has the same length as the first vector.

Usage

```
do_align(y1, y2, min.overlap = 0.1 * max(length(y1), length(y2)))
```

Arguments

y1	vector of striation marks (assuming equidistance between values)
y2	vector of striation marks
min.overlap	integer value: what is the minimal number of values between y1 and y2 that should be considered?

Value

list consisting of a) the maximal cross correlation, b) the lag resulting in the highest cross correlation, and c) a vector of length y1 with aligned values of y2.

Examples

```
library(dplyr)
x <- runif(20)
do_align(x, lead(x, 5))
do_align(x, lag(x, 5), min.overlap=2)
do_align(x, lag(x, 5), min.overlap=3)
do_align(x, x[-(1:5)], min.overlap=3)
do_align(x[-(1:5)], x, min.overlap=3)
```

fit_loess *Fit a loess curve to a bullet data frame*

Description

First, the surface measurements of the bullet land is trimmed to be within left and right groove as specified by vector groove. A loess regression is fit to the remaining surface measurements and residuals are calculated. The most extreme 0.25 The result is called the signature of the bullet land.

Usage

```
fit_loess(bullet, groove, span = 0.75)
```

Arguments

bullet	The bullet object as returned from x3p_to_df
groove	vector of two numeric values indicating the location of the left and right groove.
span	The span to use for the loess regression

Value

a list of a data frame of the original bullet measurements extended by loess fit, residuals, and standard errors and two plots: a plot of the fit, and a plot of the bullet's land signature.

---

fortify_x3p *Convert an x3p file into a data frame*

Description

old function - for previous ISO standard. x3p format consists of a list with header info and a 2d matrix of scan depths. fortify_x3p turn the matrix into a variable within a data frame, using the parameters of the header as necessary.

Usage

```
fortify_x3p(x3p)
```

Arguments

x3p	a file in x3p format as return by function read_x3p

Value

data frame with variables x, y, and value

Examples

```
data(br411)
br411_fort <- fortify_x3p(br411)
head(br411_fort)
```

getCircle *Estimate center and radius*

Description

Assuming the variables x and y are describing points located on a circle, the function uses a likelihood approach to estimate center and radius of the circle.

Usage

```
getCircle(x, y)
```

Arguments

x	numeric vector of values
y	numeric vector of values

Value

three dimensional vector of the circle center (x0, y0) and the radius

getTwist *Estimate the twist in a bullet land*

Description

Estimation of the twist in a barrel follows roughly the process described by Chu et al (2010). At the moment, twist is estimated from a single land - but the twist should be the same for the whole barrel. Therefore all lands of the same barrel should have the same twist. A note on timing: at the moment calculating the twist rate for a bullet land takes several minutes. XXX TODO XXX make the different methods a parameter. Also, accept other input than the path - if we start with the flattened bulletland we get results much faster.

Usage

```
getTwist(path, bullet = NULL, twistlimit = NULL, cutoff = 0.75)
```

Arguments

path	to a file in x3p format
bullet	data in x3p format as returned by function read_x3p
twistlimit	Constraint the possible twist value
cutoff	Use this for the quantile cutoff

Value

numeric value estimating the twist

Examples

```
## Not run:
# execution takes several minutes
load("data/b1.rda")
twist <- getTwist(path="barrel 1 bullet 1", bullet = b1, twistlimit=c(-2,0)*1.5625)

## End(Not run)
```

---

get_bullet *Deprecated function use get_crosscut*

Description

Deprecated function use get_crosscut

Usage

```
get_bullet(path, x = 243.75)
```

Arguments

path	The path to the x3p file
x	The crosscut value

---

get_chumbley *Compute a Chumbley test score*

Description

Compute a Chumbley test score

Usage

```
get_chumbley(y1, y2, window, reps = 3)
```

Arguments

window	width of the window (in indices) to consider for matching
reps	number of replicates to use in the evaluation
b1	vector of equi-distant toolmark values
b2	vector of equi-distant toolmark values

References

Chumbley, L. S., Morris, M. D., Kreiser, M. J., Fisher, C., Craft, J., Genalo, L. J., Davis, S., Faden, D. and Kidd, J. (2010), Validation of Tool Mark Comparisons Obtained Using a Quantitative, Comparative, Statistical Algorithm. Journal of Forensic Sciences, 55: 953<e2><80><93>961. doi:10.1111/j.1556-4029.2010.01424.x

Examples

```
library(dplyr)

data(br411)
b1 <- get_crosscut(x = 250, bullet=br411)
b2 <- get_crosscut(x = 150, bullet = br411)
b3 <- get_crosscut(x = 10, bullet=br411)
b1.gr <- b1 %>% get_grooves(smoothfactor=30)
b2.gr <- b2 %>% get_grooves()
b3.gr <- b3 %>% get_grooves()
# check that the grooves are actually found:
b1.gr$plot
b2.gr$plot
# get signatures
b1 <- fit_loess(b1, b1.gr)$data
b2 <- fit_loess(b2, b2.gr)$data
b3 <- fit_loess(b3, b3.gr)$data
get_chumbley(b1$resid, b2$resid, window=150, reps=5)
get_chumbley(b1$resid, b2$resid, window=150, reps=5)
get_chumbley(b1$resid, b2$resid, window=50, reps=12)
```

`get_cor` *Get correlation between two signatures*

Description

Get correlation between two signatures

Usage

```
get_cor(b1, b2, window, b1.left, lag)
```

Arguments

`b1`	dataframe
`b2`	dataframe
`window`	width of the window (in indices) to consider for matching
`b1.left`	left index location of the matching window
`lag`	integer lag for the second window

get_crosscut *Read a crosscut from a 3d surface file*

Description

Read a crosscut from a 3d surface file

Usage

```
get_crosscut(path = NULL, x = 243.75, bullet = NULL)
```

Arguments

path	path to an x3p file. The path will only be considered, if bullet is not specified.
x	level of the crosscut to be taken. If this level does not exist, the crosscut with the closest level is returned.
bullet	alternative access to the surface measurements.

Value

data frame

get_D *Compute the Euclidean distance between toolmarks*

Description

Compute the Euclidean distance between two toolmark patterns. The striation patterns are not aligned before the distance is calculated.

Usage

```
get_D(y1, y2, normalize = TRUE, resolution = 0.645)
```

Arguments

y1	vector of equi-distant toolmark values
y2	vector of equi-distant toolmark values
normalize	should the result be normalized to 1000 microns (1 millimeter)? Defaults to TRUE.
resolution	microns per pixel. Only used for normalization.

get_features *Get a feature vector for a pair of lands*

Description

Get a feature vector for a pair of lands

Usage

```
get_features(res)
```

Arguments

res	list of two aligned lands resulting from bulletGetMaxCMS

get_grooves *Find the grooves of a bullet land*

Description

Find the grooves of a bullet land

Usage

```
get_grooves(bullet, method = "rollapply", smoothfactor = 15, adjust = 10,
  groove_cutoff = 400, mean_left = NULL, mean_right = NULL,
  mean_window = 100)
```

Arguments

bullet	data frame with topological data in x-y-z format
method	method to use for identifying grooves. Defaults to "rollapply"
smoothfactor	The smoothing window to use - XXX the smoothing window seems to depend on the resolution at which the data has been collected.
adjust	positive number to adjust the grooves - XXX should be expressed in microns rather than an index
groove_cutoff	The index at which a groove cannot exist past - XXX this parameter should be expressed in microns rather than as an index to be able to properly deal with different resolutions
mean_left	If provided, the location of the average left groove
mean_right	If provided, the location of the average right groove
mean_window	The window around the means to use
second_smooth	Whether or not to smooth a second time

get_grooves_middle *Use the center of a crosscut*

Description

Use the center of a crosscut

Usage

```
get_grooves_middle(bullet, middle = 75)
```

Arguments

bullet	data frame with topological data in x-y-z format
middle	middle percent to use for the identification

get_grooves_quadratic *Quadratic fit to find groove locations*

Description

Use a robust fit of a quadratic curve to find groove locations

Usage

```
get_grooves_quadratic(bullet, adjust)
```

Arguments

bullet	data frame with topological data in x-y-z format
adjust	positive number to adjust the grooves

get_grooves_rollapply *Using rollapply to find grooves in a crosscut*

Description

Using rollapply to find grooves in a crosscut

Usage

```
get_grooves_rollapply(bullet, smoothfactor = 15, adjust = 10,
  groove_cutoff = 400, mean_left = NULL, mean_right = NULL,
  mean_window = 100, second_smooth = T, which_fun = mean)
```

Arguments

bullet	data frame with topological data in x-y-z format
smoothfactor	The smoothing window to use
adjust	positive number to adjust the grooves
groove_cutoff	The index at which a groove cannot exist past
mean_left	If provided, the location of the average left groove
mean_right	If provided, the location of the average right groove
mean_window	The window around the means to use
second_smooth	Whether or not to smooth a second time
which_fun	Which function to use in the rollapply statement

get_H *Compute the Hausdorff distance between toolmarks*

Description

Compute the Housdorff distance between two toolmark patterns. The striation patterns are not aligned before the distance is calculated. The Hausdorff distance is defined as the maximum among the shortest distances between two curves. Here, we allow to trim the largest distances to make the distance more robust

Usage

```
get_H(y1, y2, trim = 0)
```

Arguments

y1	vector of equi-distant toolmark values
y2	vector of equi-distant toolmark values
trim	percentage of largest distances to be trimmed.

get_lag — *Get best lag for two vectors based on cross-correlation*

Description

A small piece (b2) is matched to a much larger piece (b1). The lag gives the index location of the best match of b2 in b1. This function is essentially just a wrapper for `my_ccf`, but adds a plot of the result for convenience.

Usage

```
get_lag(b1, b2, negperc = 10)
```

Arguments

b1	vector of striation marks (assuming equidistance between values)
b2	(smaller) vector of striation marks
negperc	amount of lead that b2 can have compared to b1

Value

list of lag and correlation achieved. The plot shows b2 on b1

get_lag_max_R — *Get R Statistic for Chumbley matching*

Description

See Chumbley et al (2010). A small piece (b2) is matched to a much larger piece (b1). The lag gives the index location of the best match of b2 in b1.

Usage

```
get_lag_max_R(b1, b2, window, b1.left)
```

Arguments

b1	dataframe
b2	dataframe
window	width of the window (in indices) to consider for matching
b1.left	left index of the matching window

Value

list of lag and correlation achieved. The plot shows b2 on b1.

get_peaks *Identify the location and the depth of peaks and heights at a crosscut*

Description

Identify the location and the depth of peaks and heights at a crosscut

Usage

```
get_peaks(loessdata, column = "resid", smoothfactor = 35, striae = TRUE,
  window = TRUE)
```

Arguments

loessdata	export from rollapply
column	The column which should be smoothed
smoothfactor	set to default of 35. Smaller values will pick up on smaller changes in the crosscut.
striae	If TRUE, show the detected striae on the plot
window	If TRUE, show the window of the striae on the plot

Value

list of several objects:

get_peaks_nist *Identify the location and the depth of peaks and heights at a crosscut*

Description

Identify the location and the depth of peaks and heights at a crosscut

Usage

```
get_peaks_nist(loessdata, column = "resid", smoothfactor = 35,
  striae = TRUE, window = TRUE)
```

Arguments

loessdata	export from rollapply
column	The column which should be smoothed
smoothfactor	set to default of 35. Smaller values will pick up on smaller changes in the crosscut.
striae	If TRUE, show the detected striae on the plot
window	If TRUE, show the window of the striae on the plot

Value

list of several objects:

`maxCMS` *Number of maximum consecutively matching striae*

Description

Number of maximum consecutively matching striae

Usage

```
maxCMS(match)
```

Arguments

`match`	is a Boolean vector of matches/non-matches

Value

an integer value of the maximum number of consecutive matches

Examples

```
x <- rbinom(100, size = 1, prob = 1/3)
CMS(x == 1) # expected value for longest match is 3
maxCMS(x==1)
```

`my_ccf` *Cross correlation function between two vectors*

Description

Cross correlation function between two vectors

Usage

```
my_ccf(x, y, min.overlap = 0.1 * max(length(x), length(y)))
```

Arguments

`x`	vector
`y`	vector
`min.overlap`	integer value: what is the minimal number of values between x and y that should be considered?

Value

list with ccf values and lags

Examples

```
library(dplyr)
x <- runif(20)
my_ccf(x, lead(x, 5))
my_ccf(x, lag(x, 5), min.overlap=3)
x <- runif(100)
my_ccf(x[45:50], x, min.overlap=6)
```

---

plot_3d_land *Plot a bullet land using plotly*

Description

Plot a bullet land using plotly

Usage

```
plot_3d_land(path, bullet = NULL, sample = 1, ...)
```

Arguments

path	The path to the x3p file
bullet	If not null, use this pre-loaded bullet
sample	integer value. take every 1 in sample values from the surface matrix
...	parameters passed on to plot_ly call

Examples

```
data(br411)
plot_3d_land(bullet=br411, sample=2)
```

predCircle *Estimate predictions and residuals for a circle fit of x and y*

Description

estimate a circle, find predictive values and resiudals. depending on specification, vertical (regular) residuals or orthogonal residuals are computed.

Usage

```
predCircle(x, y, resid.method = "response")
```

Arguments

x	vector of numeric values
y	vector of numeric values
resid.method	character, one of "response" or "ortho"(gonal)

Value

data frame with predictions and residuals

predSmooth *Estimate predictions and residuals for a smooth of x and y*

Description

Fit a smooth line throught x and y, find predictive values and resiudals.

Usage

```
predSmooth(x, y)
```

Arguments

x	vector of numeric values
y	vector of numeric values

Value

data frame with predictions and residuals

`processBullets` *Process x3p file*

Description

x3p file of a 3d topological bullet surface is processed at surface crosscut x, the bullet grooves in the crosscuts are identified and removed, and a loess smooth is used (see `?loess` for details) to remove the big structure.

x3p file of a 3d topological bullet surface is processed at surface crosscut x, the bullet grooves in the crosscuts are identified and removed, and a loess smooth is used (see `?loess` for details) to remove the big structure.

Usage

```
processBullets(bullet, name = "", x = 100, grooves = NULL, span = 0.75,
  window = 1, ...)

processBullets(bullet, name = "", x = 100, grooves = NULL, span = 0.75,
  window = 1, ...)
```

Arguments

`bullet`	file as returned from read_x3p
`name`	name of the bullet
`x`	(vector) of surface crosscuts to process (in meters).
`grooves`	The grooves to use as a two element vector, if desired
`span`	The span for the loess fit
`window`	The mean window around the ideal crosscut
`...`	Additional arguments, passed to the get_grooves function
`bullet`	file as returned from read_x3p
`name`	name of the bullet
`x`	(vector) of surface crosscuts to process.
`grooves`	The grooves to use as a two element vector, if desired
`span`	The span for the loess fit
`window`	The window around the ideal crosscut
`...`	Additional arguments, passed to the get_grooves function

Value

data frame

data frame

Examples

```
data(br411)
br411_processed <- processBullets(br411, name = "br411")
data(br411)
br411_processed <- processBullets(br411, name = "br411")
```

`read_dat` *Read a dat file and create an x3p file*

Description

Read a dat file and create an x3p file

Usage

```
read_dat(path, profiley = TRUE, sample = 1)
```

Arguments

`path`	The file path to the dat file
`profiley`	are profiles on y?
`sample`	1 in sample lines will be taken

Format

list with header information and surface matrix

Examples

```
## Not run:
d1 <- read_dat("Br4 Bullet 4-1.dat")
d2 <- read_dat("L1.dat", profiley = FALSE)

## End(Not run)
```

rtrees *randomforest*

Description

this randomforest was fitted to predict known matches and non-matches from the scans of land engraved areas of the Hamby study.

Usage

```
rtrees
```

Format

a random forest object fitted by the randomforest function from the package of the same name

smoothloess *Predict smooth from a fit*

Description

Predict smooth from a fit

Usage

```
smoothloess(x, y, span, sub = 2)
```

Arguments

x	X values to use
y	Y values to use
span	The span of the loess fit
sub	Subsample factor

striation_identify *Match striation marks across two cross sections based on previously identified peaks and valleys*

Description

Match striation marks across two cross sections based on previously identified peaks and valleys

Usage

```
striation_identify(lines1, lines2)
```

Arguments

lines1	data frame as returned from get_peaks function. data frames are expected to have the following variables: xmin, xmax, group, type, bullet, heights
lines2	data frame as returned from get_peaks function. data frames are expected to have the following variables: xmin, xmax, group, type, bullet, heights

Value

data frame of the same form as lines1 and lines2, but consisting of an additional variable of whether the striation marks are matches

unfortify_x3p *Convert a data frame into an x3p file*

Description

Convert a data frame into an x3p file

Usage

```
unfortify_x3p(df)
```

Arguments

df	A data frame produced by fortify_x3p

Value

An x3p object

Examples

```
data(br411)
br411_fort <- fortify_x3p(br411)
br411_unfort <- unfortify_x3p(br411_fort)
identical(br411_unfort, br411)
```

Index