# Package 'cmpsR'

October 12, 2022

**Title** R Implementation of Congruent Matching Profile Segments Method

**Version** 0.1.2

**Description** This is an open-source implementation of the Congruent Matching Profile
Segments (CMPS) method (Chen et al. 2019)<doi:10.1016/j.forsciint.2019.109964>.
In general, it can be used for objective comparison of striated tool marks, and in our
examples, we specifically use it for bullet signatures comparisons. The CMPS score is
expected to be large if two signatures are similar. So it can also be considered as a
feature that measures the similarity of two bullet signatures.

**Imports** assertthat (>= 0.2.0), dplyr (>= 1.0.5), rlang (>= 0.4.5),
ggplot2 (>= 3.3.0)

**Suggests** purrr, tidyverse, ggpubr, knitr, rmarkdown

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.2

**Depends** R (>= 3.5.0)

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Wangqian Ju [aut, cre] (<https://orcid.org/0000-0002-9977-377X>),
Heike Hofmann [ctb] (<https://orcid.org/0000-0001-6216-5183>)

**Maintainer** Wangqian Ju <wju@iastate.edu>

**Repository** CRAN

**Date/Publication** 2022-07-18 08:20:05 UTC

## R topics documented:

---

bullets                                    *Information of two example bullets*

---

### Description

A dataset containing pre-processed information of two bullets. They are used as examples in Chapter 3.5 of Open Forensic Science in R.

### Usage

```
bullets
```

### Format

A data frame/tbl/tbl_df with 12 rows and 3 variables:

**source**  source of the bullet data

**sigs**  bullet signatures, detailed information about how to get the signatures can be found at https://sctyner.github.io/OpenForS

**bulletland**  label of the signatures

### Source

<https://sctyner.github.io/OpenForSciR/bullets.html>

---

| cmps_na_trim | *Remove the leading and trailing missing values in a numeric vector* |
|---|---|

---

### Description

Remove the leading and trailing missing values in a numeric vector

### Usage

```
cmps_na_trim(x)
```

### Arguments

x                    numeric vector

### Value

a numeric vector; only the leading and trailing missing values are removed

### Examples

```
x <- c(NA, 1, 2, 3, 4, NA)
cmps_na_trim(x)
```

---

| cmps_segment_plot | *Plot the selected basis segment and its cross-correlation curve at all scales based on the results of CMPS algorithm* |
|---|---|

---

### Description

This function plots the selected basis segment with the comparison signature. One can visualize the scaled segment and its corresponding cross-correlation curve. The number of marked correlation peaks at each segment scale is determined by npeaks_set of extract_feature_cmps. The red vertical dashed line indicates the congruent registration position for all segments; the green vertical dashed line indicates the position of the consistent correlation peak (if any); the blue vertical dashed line indicates the tolerance zone (determined by Tx)

### Usage

```
cmps_segment_plot(cmps_result, seg_idx = 1)
```

## Arguments

cmps_result    a list generated by extract_feature_cmps. cmps_result is required to have
               the following names: parameters, congruent_pos, segments, nseg, i.e. one
               should at least have include = c("parameters", "congruent_pos", "segments",
               "nseg") when computing cmps_result. However, include = "full_result
               is still recommended.

seg_idx        an integer. The index of a basis segment that we want to plot for.

## Value

a list of n elements, where n is the length of npeaks_set, i.e. the number of scales for each basis
segment. And each one of these n elements is also a list, a list of two plots:

- segment_plot: The basis segment of current scale is plotted at different positions where the
  segment obtains correlation peak. The comparison signature is also plotted.

- scale_ccf_plot: This is the plot of the cross-correlation curve between the comparison sig-
  nature and the segment of the current scale.

## Examples

```
library(cmpsR)
library(ggpubr)

data("bullets")
land2_3 <- bullets$sigs[bullets$bulletland == "2-3"][[1]]
land1_2 <- bullets$sigs[bullets$bulletland == "1-2"][[1]]

# compute cmps

# algorithm with multi-peak insepction at three different segment scales
cmps_with_multi_scale <- extract_feature_cmps(land2_3$sig, land1_2$sig, include = "full_result" )

# generate plots using cmps_signature_plot
seg_plot <- cmps_segment_plot(cmps_with_multi_scale, seg_idx = 3)

pp <- ggarrange(plotlist = unlist(seg_plot, recursive = FALSE), nrow = 3, ncol = 2)
```

---

cmps_signature_plot    *Plot reference signature and comparison signature based on the re-
                        sults of CMPS algorithm*

---

## Description

This function aligns two signatures and shows which basis segments find the congruent registration
position.

## Usage

```
cmps_signature_plot(cmps_result, add_background = TRUE)
```

## Arguments

cmps_result    a list generated by extract_feature_cmps. cmps_result is required to have
               the following names: parameters, congruent_seg, congruent_pos, segments,
               ccp_list. So include = "full_result" is recommended when computing
               cmps_result

add_background boolean; whether or not to add zebra-striped background under each basis seg-
               ment; default is TRUE

## Value

a list

- segment_shift_plot: a plot object generated by ggplot2. In this plot only basis segments
  that are congruent matching profile segments (CMPS) are plotted along with the compari-
  son profile; each basis segment is shifted to the position where it obtains either a consistent
  correlation peak or a cross-correlation peak closest to the congruent registration position

- signature_shift_plot: a plot object generated by ggplot2. In this plot both the reference
  signature and the comparison signature are plotted, and CMPS are highlighted. The align-
  ment of the two signatures is achieved by shifting the reference signature to the congruent
  registration position.

- seg_shift: a data.frame. This data frame shows which basis segments are plotted (are
  CMPS) and the units by which each segment shifted when plotting segment_shift_plot

- sig_shift: a numeric value. The number of units by which the reference signature shifted
  when plotting signature_shift_plot

## Examples

```
library(cmpsR)

data("bullets")
land2_3 <- bullets$sigs[bullets$bulletland == "2-3"][[1]]
land1_2 <- bullets$sigs[bullets$bulletland == "1-2"][[1]]

# compute cmps

# algorithm with multi-peak insepction at three different segment scales
cmps_with_multi_scale <- extract_feature_cmps(land2_3$sig, land1_2$sig, include = "full_result" )

# generate plots using cmps_signature_plot
sig_plot <- cmps_signature_plot(cmps_with_multi_scale)
```

---

| compute_cross_corr | *Wrapper function for compute_cross_corr* |

---

### Description

Wrapper function for compute_cross_corr

### Usage

```
compute_cross_corr(x, y, min.overlap)
```

### Arguments

| | |
|---|---|
| x | numeric vector, the longer sequence |
| y | numeric vector, the shorter sequence |
| min.overlap | numeric scalor, set the length of the minimum overlapping part |

---

| compute_diff_phase | *Compute a Statistic for the Foreground Phase and the Background Phases* |

---

### Description

Compute a statistic (for example, a mean) based on all matching comparisons (foreground phase) and the same statistic based on all non-matching comparisons (background phases)

### Usage

```
compute_diff_phase(scores_list, FUNC = mean, na.rm = TRUE, both = FALSE)
```

### Arguments

| | |
|---|---|
| scores_list | a list of all phases |
| FUNC | a function to be applied to both the foreground phase and the background phases |
| na.rm | a logical value indicating whether NA values should be stripped before the computation proceeds |
| both | logical value. If TRUE, return the values of the FUNC for both the foreground phase and the background phases; if FALSE, return their difference |

### Value

If both = TRUE, return the values of the statistic (calculated by FUNC) for both the foreground phase and the background phases; if both = FALSE, return the difference

**Examples**

```
library(tidyverse)

data("bullets")

lands <- unique(bullets$bulletland)

comparisons <- data.frame(expand.grid(land1 = lands[1:6], land2 = lands[7:12]),
                          stringsAsFactors = FALSE)

comparisons <- comparisons %>%
  left_join(bullets %>% select(bulletland, sig1=sigs),
            by = c("land1" = "bulletland")) %>%
  left_join(bullets %>% select(bulletland, sig2=sigs),
            by = c("land2" = "bulletland"))

comparisons <- comparisons %>% mutate(
  cmps = purrr::map2(sig1, sig2, .f = function(x, y) {
    extract_feature_cmps(x$sig, y$sig, include = "full_result")
  })
)

comparisons <- comparisons %>%
  mutate(
    cmps_score = sapply(comparisons$cmps, function(x) x$CMPS_score),
    cmps_nseg = sapply(comparisons$cmps, function(x) x$nseg)
  )

cp1 <- comparisons %>% select(land1, land2, cmps_score, cmps_nseg)
cp1 <- cp1 %>% mutate(
  land1idx = land1 %>% str_sub(-1, -1) %>% as.numeric(),
  land2idx = land2 %>% str_sub(-1, -1) %>% as.numeric()
)

phases <- with(cp1, {
  get_all_phases(land1idx, land2idx, cmps_score, addNA = TRUE)
})

compute_diff_phase(phases)
```

---

compute_score_metrics    *Compute Different Metrics Based on Scores*

---

**Description**

Compute Different Metrics Based on Scores

**Usage**

```
compute_score_metrics(
  land1,
  land2,
  score,
  addNA = TRUE,
  na.rm = TRUE,
  include = NULL,
  out_names = NULL
)
```

**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 into a single number |
| addNA | logical value. In case of missing lands, are scores set to 0 (addNA = FALSE) or set to NA (addNA = TRUE) |
| na.rm | a logical value indicating whether NA values should be stripped before the computation proceeds |
| include | a character vector specifying which metrics to be included in the result; if include = NULL, including all metrics |
| out_names | a character vector specifying the variable names of each metric; if out_names = NULL, using the default names |

**Details**

By default, this helper function computes four metrics.

`diff`: the difference between the mean score of the foreground phase and the mean score of the background phases `diff.med`: the difference between the median score of the foreground phase and the median score of the background phases `max`: the max score `maxbar`: the mean score of the foreground phase

**Value**

a data frame containing values of the metrics

**Examples**

```
library(tidyverse)

data("bullets")

lands <- unique(bullets$bulletland)

comparisons <- data.frame(expand.grid(land1 = lands[1:6], land2 = lands[7:12]),
                          stringsAsFactors = FALSE)
```

```
comparisons <- comparisons %>%
  left_join(bullets %>% select(bulletland, sig1=sigs),
            by = c("land1" = "bulletland")) %>%
  left_join(bullets %>% select(bulletland, sig2=sigs),
            by = c("land2" = "bulletland"))

comparisons <- comparisons %>% mutate(
  cmps = purrr::map2(sig1, sig2, .f = function(x, y) {
    extract_feature_cmps(x$sig, y$sig, include = "full_result")
  })
)

comparisons <- comparisons %>%
  mutate(
    cmps_score = sapply(comparisons$cmps, function(x) x$CMPS_score),
    cmps_nseg = sapply(comparisons$cmps, function(x) x$nseg)
  )

cp1 <- comparisons %>% select(land1, land2, cmps_score, cmps_nseg)
cp1 <- cp1 %>% mutate(
  land1idx = land1 %>% str_sub(-1, -1) %>% as.numeric(),
  land2idx = land2 %>% str_sub(-1, -1) %>% as.numeric()
)

with(cp1, {
  compute_score_metrics(land1idx, land2idx, cmps_score)
})
```

---

compute_ss_ratio          *#' Compute the Sum of Squares Ratio*

---

### Description

#' Compute the Sum of Squares Ratio

### Usage

```
compute_ss_ratio(score, label, MS = FALSE)
```

### Arguments

| | |
|---|---|
| score | a numeric vector, scores |
| label | a character vector, the label of each score |
| MS | boolean, whether to compute the mean squares instead of the sum of squares. Default is FALSE |

### Value

the sum of squares ratio

## Examples

```
score <- c(rnorm(100), rnorm(100, mean = 5))
label <- c(rep("a", 100), rep("b", 100))
compute_ss_ratio(score, label)
```

---

extract_feature_cmps        *Computes the CMPS score of a comparison between two bullet pro-*
                            *files/signatures*

---

## Description

Compute the Congruent Matching Profile Segments (CMPS) score based on two bullet profiles/signatures.
The reference profile will be divided into consecutive, non-overlapping, basis segments of the same
length. Then the number of segments that are congruent matching will be found as the CMPS score.
By default, extract_feature_cmps implements the algorithm with multi-peak inspection at three
different segment scale levels. By setting npeaks_set as a single-length vector, users can switch to
the algorithm with multi-peak inspection at the basis scale level only.

## Usage

```
extract_feature_cmps(
  x,
  y,
  seg_length = 50,
  Tx = 25,
  npeaks_set = c(5, 3, 1),
  include = NULL,
  outlength = NULL
)
```

## Arguments

| | |
|---|---|
| x | a numeric vector, vector of the reference bullet signature/profile that will be divided into basis segments |
| y | a numeric vector, vector of the comparison bullet signature/profile |
| seg_length | a positive integer, the length of a basis segment |
| Tx | a positive integer, the tolerance zone is +/- Tx |
| npeaks_set | a numeric vector, specify the number of peaks to be found at each segment scale level |

  - If length(npeaks_set) == 1, the algorithm uses multi-peak inspection only
    at the basis scale level;
  - If length(npeaks_set) > 1, the algorithm uses multi-peak inspection at
    different segment scale levels.
  - By default, npeaks_set = c(5,3,1). Including more segment scale levels
    will reduce the number of false positive results

include  NULL or a vector of character strings indicating what additional information should be included in the output of extract_feature_cmps. All possible options are: "nseg", "congruent_pos", "congruent_seg", "congruent_seg_idx", "pos_df", "ccp_list","segments", and "parameters". If one wants to include them all, one can use include = "full_result". By default, include = NULL and only the CMPS score is returned

outlength  NULL or a numeric vector, specify the segment length of each level of the basis segment when the multi-segment lengths strategy is being used. If outlength = NULL, then the length of a basis segment will be doubled at each segment level

**Value**

a numeric value or a list

- if include = NULL, returns the CMPS score (a numeric value) only

- if include = one or a vector of strings listed above:

    - nseg: number of basis segments
    - congruent_seg: a vector of boolean values. TRUE means this basis segment is a congruent matching profile segment (CMPS)
    - congruent_seg_idx: the indices of all CMPS
    - pos_df: a dataframe that includes positions of correlation peaks and the CMPS score of these positions
    - ccp_list: a list of consistent correlation peaks of each basis segment.
    - segments: a list of all basis segments
    - parameters: a list that stores all parameters used in the function call

**References**

Chen, Zhe, Wei Chu, Johannes A Soons, Robert M Thompson, John Song, and Xuezeng Zhao. 2019. "Fired Bullet Signature Correlation Using the Congruent Matching Profile Segments (CMPS) Method." Forensic Science International, December, #109964. https://doi.org/10.1016/j.forsciint.2019.109964.

**Examples**

```
library(tidyverse)
library(cmpsR)

data("bullets")
land2_3 <- bullets$sigs[bullets$bulletland == "2-3"][[1]]
land1_2 <- bullets$sigs[bullets$bulletland == "1-2"][[1]]

# compute cmps

# algorithm with multi-peak insepction at three different segment scale levels
cmps_with_multi_scale <- extract_feature_cmps(land2_3$sig, land1_2$sig, include = "full_result" )

# algorithm with multi-peak inspection at the basis scale level only
cmps_without_multi_scale <- extract_feature_cmps(land2_3$sig, land1_2$sig,
                                                 npeaks_set = 5, include = "full_result" )
```

```
# Another example
library(tidyverse)

data("bullets")

lands <- unique(bullets$bulletland)

comparisons <- data.frame(expand.grid(land1 = lands[1:6], land2 = lands[7:12]),
                          stringsAsFactors = FALSE)

comparisons <- comparisons %>%
  left_join(bullets %>% select(bulletland, sig1=sigs),
            by = c("land1" = "bulletland")) %>%
  left_join(bullets %>% select(bulletland, sig2=sigs),
            by = c("land2" = "bulletland"))

comparisons <- comparisons %>% mutate(
  cmps = purrr::map2(sig1, sig2, .f = function(x, y) {
    extract_feature_cmps(x$sig, y$sig, include = "full_result")
  })
)

comparisons <- comparisons %>%
  mutate(
    cmps_score = sapply(comparisons$cmps, function(x) x$CMPS_score),
    cmps_nseg = sapply(comparisons$cmps, function(x) x$nseg)
  )

cp1 <- comparisons %>% select(land1, land2, cmps_score, cmps_nseg)
cp1
```

---

get_all_phases         *Obtain a list of all phases of a bullet-by-bullet comparison*

---

## Description

Obtain a list of all phases of a bullet-by-bullet comparison

## Usage

```
get_all_phases(land1, land2, score, addNA = FALSE)
```

## 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 into a single number |
| addNA | logical value. In case of missing lands, are scores set to 0 (addNA = FALSE) or set to NA (addNA = TRUE) |

**Value**

a list of all phases

**Examples**

```
library(tidyverse)

data("bullets")

lands <- unique(bullets$bulletland)

comparisons <- data.frame(expand.grid(land1 = lands[1:6], land2 = lands[7:12]),
                          stringsAsFactors = FALSE)

comparisons <- comparisons %>%
  left_join(bullets %>% select(bulletland, sig1=sigs),
            by = c("land1" = "bulletland")) %>%
  left_join(bullets %>% select(bulletland, sig2=sigs),
            by = c("land2" = "bulletland"))

comparisons <- comparisons %>% mutate(
  cmps = purrr::map2(sig1, sig2, .f = function(x, y) {
    extract_feature_cmps(x$sig, y$sig, include = "full_result")
  })
)

comparisons <- comparisons %>%
  mutate(
    cmps_score = sapply(comparisons$cmps, function(x) x$CMPS_score),
    cmps_nseg = sapply(comparisons$cmps, function(x) x$nseg)
  )

cp1 <- comparisons %>% select(land1, land2, cmps_score, cmps_nseg)
cp1 <- cp1 %>% mutate(
  land1idx = land1 %>% str_sub(-1, -1) %>% as.numeric(),
  land2idx = land2 %>% str_sub(-1, -1) %>% as.numeric()
)

with(cp1, {
  get_all_phases(land1idx, land2idx, cmps_score, addNA = TRUE)
})
```

---

get_ccf4                      *Function to calculate the cross-correlation between two sequences*

---

**Description**

This function is used for CMPS algorithm.

## Usage

```
get_ccf4(x, y, min.overlap = round(0.1 * max(length(x), length(y))))
```

## Arguments

| | |
|---|---|
| x | numeric sequence of values |
| y | numeric sequence of values |
| min.overlap | integer, minimal number of values in the overlap between sequences x and y to calculate a correlation value. Set to 10 percent of the maximum length of either sequence (HH: this might be problematic for CMPS) |

## Value

list consisting of the lag where the maximum correlation is achieved, and the maximum correlation value.

## Examples

```
data("bullets")
land2_3 <- bullets$sigs[bullets$bulletland == "2-3"][[1]]
land1_2 <- bullets$sigs[bullets$bulletland == "1-2"][[1]]
x <- land2_3$sig
y <- land1_2$sig

segments <- get_segs(x, len = 50)

ccr <- get_ccf4(y, segments$segs[[7]],
                min.overlap = length(segments$segs[[7]]))
```

---

get_ccp                          *Identify at most one consistent correlation peak (ccp)*

---

## Description

If multi segment lengths strategy is being used, at most one consistent correlation peak (ccp) will be found for the corresponding basis segment. If the ccp cannot be identified, return NULL

## Usage

```
get_ccp(ccr_list, Tx = 25)
```

## Arguments

| | |
|---|---|
| ccr_list | list, obtained by get_ccr_peaks |
| Tx | integer, the tolerance zone is +/- Tx |

## Value

integer, the position of the ccp if it is identified; NULL otherwise.

## Examples

```
data("bullets")
land2_3 <- bullets$sigs[bullets$bulletland == "2-3"][[1]]
land1_2 <- bullets$sigs[bullets$bulletland == "1-2"][[1]]
x <- land2_3$sig
y <- land1_2$sig

segments <- get_segs(x, len = 50)

# identify the consistent correlation peak when ccf curves are computed
# based on y and segment 7 in 3 different scales;
# the number of peaks identified in each scale are 5, 3, and 1, respectively.
seg_scale_max <- 3
npeaks_set <- c(5,3,1)
outlength <- c(50, 100, 200)

ccr_list <- lapply(1:seg_scale_max, function(seg_scale) {
  get_ccr_peaks(y, segments, seg_outlength = outlength[seg_scale], nseg = 7,
  npeaks = npeaks_set[seg_scale])
})

get_ccp(ccr_list, Tx = 25)
```

---

get_ccr_peaks                    *Identify peaks of a cross correlation curve*

---

## Description

Given a comparison profile and a segment, `get_ccr_peaks` computes the cross correlation curve
and finds peaks of the curve.

## Usage

```
get_ccr_peaks(comp, segments, seg_outlength, nseg = 1, npeaks = 5)
```

## Arguments

| | |
|---|---|
| comp | a nueric vector, vector of the bullet comparison profile |
| segments | list with basis segments and their corresponding indices in the original profile, obtianed by `get_segs()` |
| seg_outlength | length of the enlarged segment |
| nseg | integer. nseg = 3: the third segment in `segments` |
| npeaks | integer. the number of peaks to be identified. |

**Value**

a list consisting of:

- `ccr`: the cross correlation curve
- `adj_pos`: indices of the curve
- `peaks_pos`: position of the identified peaks
- `peaks_heights`: the cross correlation value (height of the curve) of the peaks

**Examples**

```
data("bullets")
land2_3 <- bullets$sigs[bullets$bulletland == "2-3"][[1]]
land1_2 <- bullets$sigs[bullets$bulletland == "1-2"][[1]]
x <- land2_3$sig
y <- land1_2$sig

segments <- get_segs(x, len = 50)

# compute ccf based on y and segment 7 with scale 1, then identify 5 highest peaks
ccrpeaks <- get_ccr_peaks(y, segments = segments, seg_outlength = 50,
                          nseg = 7, npeaks = 5)
```

---

get_CMPS                          *Compute the CMPS score*

---

**Description**

Compute the CMPS score from a list of positions of (consistent) correlation peaks.

**Usage**

```
get_CMPS(input_ccp, Tx = 25)
```

**Arguments**

| | |
|---|---|
| `input_ccp` | a list of positions for (consistent) correlation peaks |
| `Tx` | integer, the tolerance zone is +/- `Tx` |

**Value**

a list of six components:

- `CMPS_score`: computed CMPS score
- `nseg`: the number of basis segments
- `congruent_pos`: the congruent position that results in the CMPS score
- `congruent_seg`: a boolean vector of the congruent matching profile segments
- `congruent_seg_idx`: the index of the congruent matching profile segments
- `pos_df`: a dataframe that includes all positions and their corresponding CMPS score

## Examples

```
data("bullets")
land2_3 <- bullets$sigs[bullets$bulletland == "2-3"][[1]]
land1_2 <- bullets$sigs[bullets$bulletland == "1-2"][[1]]
x <- land2_3$sig
y <- land1_2$sig
segments <- get_segs(x, len = 50)
nseg <- length(segments$segs)
seg_scale_max <- 3
npeaks_set <- c(5,3,1)
outlength <- c(50, 100, 200)

ccp_list <- lapply(1:nseg, function(nseg) {
 ccr_list <- lapply(1:seg_scale_max, function(seg_scale) {
   get_ccr_peaks(y, segments, seg_outlength = outlength[seg_scale],
   nseg = nseg, npeaks = npeaks_set[seg_scale])
 })

 get_ccp(ccr_list, Tx = 25)
})
cmps <- get_CMPS(ccp_list, Tx = 25)
```

---

get_segs                    *Divide a bullet signature/profile into basis segments of desired length*

---

## Description

`get_segs` divides a bullet signature/profile (a numeric vector) into consecutive, non-overlapping, basis segments of the same desired length. If the profile starts or ends with a sequence of NA (missing values), the NAs will be trimmed. If the very last segment does not have the desired length, it will be dropped.

## Usage

```
get_segs(x, len = 50)
```

## Arguments

x               a numeric vector, vector of the bullet signature/profile

len             integer: the desired length of a basis segment

## Value

list with basis segments and their corresponding indices in the profile

**Examples**

```
data("bullets")
land2_3 <- bullets$sigs[bullets$bulletland == "2-3"][[1]]
x <- land2_3$sig

segments <- get_segs(x, len = 50)
```

---

get_seg_scale                    *Change the sacle of a segment*

---

**Description**

In order to identify the congruent registration position of a basis segment, the length of the basis
segment will be doubled to compute the correlation curve. get_seg_scale computes the increased
segment, which has the same center as the basis segment.

**Usage**

```
get_seg_scale(segments, nseg, out_length)
```

**Arguments**

| | |
|---|---|
| segments | list with basis segments and their corresponding indices in the original profile, obtianed by get_segs() |
| nseg | integer. nseg = 3: increase the length of the third basis segment. |
| out_length | integer. The length of the enlarged segment |

**Value**

list consisting of

- aug_seg: the increased segment
- aug_idx: the corresponding indices in the profile

**Examples**

```
data("bullets")
land2_3 <- bullets$sigs[bullets$bulletland == "2-3"][[1]]
x <- land2_3$sig

segments <- get_segs(x, len = 50)
seg5_scale3 <- get_seg_scale(segments, nseg = 5, out_length = 50)
```

---

local_max_cmps *find local maximums*

---

### Description

find local maximums

### Usage

```
local_max_cmps(x, find_max = 0)
```

### Arguments

| | |
|---|---|
| x | numeric vector, the input sequence |
| find_max | a numeric scalor, the function finds maximums if find_max = 0 finds minimums if overwise. |

---

metric_plot_helper *Helper Function for Plotting the Distribution of a Metric*

---

### Description

Helper Function for Plotting the Distribution of a Metric

### Usage

```
metric_plot_helper(
  cmps_metric,
  metric,
  scaled = FALSE,
  SSratio = TRUE,
  plot_density = TRUE,
  ...
)
```

### Arguments

| | |
|---|---|
| cmps_metric | a data frame containing values of the metric and the labels |
| metric | string. Which metric to be plotted |
| scaled | logical value. If scaled = TRUE, the values should be within the interval of [0, 1] |
| SSratio | logical value. Whether to show the sum of squares ratio value |
| plot_density | logical value. If plot_density = TRUE, the function plots group density on the y-axis; if plot_density = FALSE, it plots the count of a certain bin. |
| ... | other arguments for plotting: breaks, binwidth, and subtitle |

## Value

a ggplot object

---

| na_trim_cmps | *Wrapper function for na_trim* |
|---|---|

---

## Description

Wrapper function for na_trim

## Usage

```
na_trim_cmps(x)
```

## Arguments

x                    numeric vector

# Index