Package ‘segmetric’

January 10, 2023

**Title**  Metrics for Assessing Segmentation Accuracy for Geospatial Data

**Version**  0.3.0

**Description**  A system that computes metrics to assess the segmentation accuracy of geospatial data. These metrics calculate the discrepancy between segmented and reference objects, and indicate the segmentation accuracy. For more details on choosing evaluation metrics, we suggest seeing Costa et al. (2018) <doi:10.1016/j.rse.2017.11.024> and Jozdani et al. (2020) <doi:10.1016/j.isprsjprs.2020.01.002>.

**License**  MIT + file LICENSE

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  7.2.3

**Imports**  magrittr, graphics, units

**Suggests**  classInt, dplyr, testthat (>= 3.0.0)

**Depends**  R (>= 2.10), sf

**BugReports**  https://github.com/michellepicoli/segmetric/issues

**URL**  https://michellepicoli.github.io/segmetric/

**NeedsCompilation**  no

**Author**  Rolf Simoes [aut] (<https://orcid.org/0000-0003-0953-4132>),
           Alber Sanchez [aut] (<https://orcid.org/0000-0001-7966-2880>),
           Michelle Picoli [aut, cre] (<https://orcid.org/0000-0001-9855-2046>)

**Maintainer**  Michelle Picoli <mipicoli@gmail.com>

**Repository**  CRAN

**Date/Publication**  2023-01-10 12:30:02 UTC

R topics documented:

segmetric-package ..........................................................  2
general_functions .........................................................  4
metric_functions ..........................................................  6
Description

Metrics for assessing segmentation accuracy for geospatial data.

Purpose

The segmetric package provides a set of metrics for the segmentation accuracy assessment (or evaluation) of geospatial data. It includes more than 20 metrics used in the literature for spatial segmentation assessment (Van Rijsbergen, 1979; Levine and Nazif, 1982; Janssen and Molenaar, 1995; Lucieer and Stein, 2002; Carleer et al., 2005; Moller et al., 2007; van Coillie et al., 2008; Costa et al., 2008; Weidner, 2008; Feitosa et al., 2010; Clinton et al., 2010; Persello and Bruzzone, 2010; Yang et al., 2014; and Zhang et al., 2015).

Extensions

The segmetric package is extensible and provides a set of functions to ease the implementation of new metrics. See ?sm_reg_metric() to find how new metrics are implemented.

Contributions

Contribution to this package could be done at segmetric’s page on GitHub: https://github.com/michellepicoli/segmetric.

Author(s)

Maintainer: Michelle Picoli <mipicoli@gmail.com> (ORCID)

Authors:

• Rolf Simoes <rolfsimoes@gmail.com> (ORCID)
• Alber Sanchez <albhasan@gmail.com> (ORCID)
References


- Jaccard, P., 1912. The distribution of the flora in the alpine zone.


- Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.,


general_functions


See Also

Useful links:

- [https://michellepicoli.github.io/segmetric/](https://michellepicoli.github.io/segmetric/)
- Report bugs at [https://github.com/michellepicoli/segmetric/issues](https://github.com/michellepicoli/segmetric/issues)

---

general_functions  General functions

Description

These functions manipulate segmetric objects.

- `sm_area()`: Return a vector of areas, one for each polygon.
- `sm_centroid()`: Return the centroids of the given polygons.
- `sm_intersection()`: Return the intersection of the given simple features.
- `sm_subset_union()`: Return the union of the given simple features.
- `sm_rbind()`: Return the merge of unique simple features.
- `sm_summarize_groups()`: Summarizes metric values by a group (either reference or segment).
- `sm_options()`: Get/Set segmetric options.
Usage

sm_area(s, order = NULL)

sm_centroid(s, order = NULL)

sm_distance(s1, s2)

sm_intersection(s1, s2, touches = TRUE)

sm_subset_union(s)

sm_rbind(...)

sm_apply_group(x, groups, fn, ...)

sm_summarize_group(x, groups, fn, ...)

sm_norm_left(x, y)

sm_norm_right(x, y)

sm_options(..., digits = NULL)

Arguments

s, s1, s2 Either a ref_sf, a seg_sf, or a subset_sf object (inherited from sf).
order A subset_sf. This argument arranges the returned values according to the ob-
ject passed here.
touches A logical. Is the border part of the intersection?
... For sm_rbind(), a set of subset_sf objects to be merged.
x, y A numeric values (e.g. areas, lengths) to compute metrics.
groups A group identifier vector used to aggregate a metric for each group.
fn A function to aggregate a metric for a group.
digits An integer indicating how many digits used to round metric values.

Value

• sm_area(): Return a numeric vector with polygons' area.
• sm_centroid(): Return a subset_sf object with polygons' centroid.
• sm_intersection(): Return a subset_sf object with intersection between polygons.
• sm_subset_union(): Return a subset_sf object with union between intersecting polygons.
• sm_rbind(): Return a subset_sf object with unique features.
Description

The `sm_compute()` computes a given metric (metric_id parameter) from segmentation objects. It compares the reference to the segmentation polygons using a metric.

A list with all supported metrics can be obtained by `sm_list_metrics()` (see Details for more information).

The `sm_metric_subset()` returns the subset used to compute the metrics in segmetric object.

Usage

```r
sm_compute(m, metric_id, ...)
sm_metric_subset(m, metric_id = NULL)
```

Arguments

- `m` A segmetric object.
- `metric_id` A character vector with metrics id to be computed.
- `...` Any additional argument to compute a metric (see Details).

Details

- "OS1" refers to Oversegmentation. Its values range from 0 (optimal) to 1 (Clinton et al., 2010).
- "US1" refers to Undersegmentation. Its values range from 0 (optimal) to 1 (Clinton et al., 2010).
- "OS2" refers to Oversegmentation. Its values range from 0 (optimal) to 1 (Persello and Bruzzone, 2010).
- "US2" refers to Undersegmentation. Its values range from 0 (optimal) to 1 (Persello and Bruzzone, 2010).
- "OS3" refers to Oversegmentation. Its values range from 0 (optimal) to 1 (Yang et al., 2014).
- "US3" refers to Undersegmentation. Its values range from 0 (optimal) to 1 (Yang et al., 2014).
- "AFI" refers to Area Fit Index. Its optimal value is 0 (Lucieer and Stein, 2002; Clinton et al., 2010).
- "QR" refers to Quality Rate. Its values range from 0 (optimal) to 1 (Weidner, 2008; Clinton et al., 2010).
- "D_index" refers to Index D. Its values range from 0 (optimal) to 1 (Levine and Nazif, 1982; Clinton et al., 2010).
- "precision" refers to Precision. Its values range from 0 to 1 (optimal) (Van Rijsbergen, 1979; Zhang et al., 2015).
metric_functions

- "recall" refers to Recall. Its values range from 0 to 1 (optimal) (Van Rijsbergen, 1979; Zhang et al., 2015).
- "UMerging" refers to Undermerging. Its values range from 0 (optimal) to 1 (Levine and Nazif, 1982; Clinton et al., 2010).
- "OMerging" refers to Overmerging. Its optimal value is 0 (Levine and Nazif, 1982; Clinton et al., 2010).
- "M" refers to Match. Its values range from 0 to 1 (optimal) (Janssen and Molenaar, 1995; Feitosa et al., 2010).
- "E" refers to Evaluation Measure. Its values range from 0 (optimal) to 100 (Carleer et al., 2005).
- "RAsub" refers to Relative Area. Its values range from 0 to 1 (optimal) (Müller et al., 2007; Clinton et al., 2010).
- "RAsuper" refers to Relative area. Its values range from 0 to 1 (optimal) (Müller et al., 2007; Clinton et al., 2010).
- "PI" refers to Purity Index. Its values range from 0 to 1 (optimal) (van Coillie et al., 2008).
- "Fitness" refers to Fitness Function. Its optimal value is 0 (Costa et al., 2008).
- "ED3" refers to Euclidean Distance. Its values range from 0 (optimal) to 1 (Yang et al., 2014).
- "F_measure" refers to F-measure metric. Its values range from 0 to 1 (optimal) (Van Rijsbergen, 1979; Zhang et al., 2015). It takes the optional weight argument alpha, ranging from 0.0 to 1.0 (the default is 0.5).
- "IoU" refers to Intersection over Union metric. Its values range from 0 to 1 (optimal) (Jaccard, 1912; Rezatofighi et al., 2019).
- "SimSize" refers to the similarity size metric. Its values range from 0 to 1 (optimal) (Zhan et al., 2005).
- "qLoc" refers to quality of object’s location metric. Its optimal value is 0 (Zhan et al., 2005).
- "RPsub" refers to Relative Position (sub) metric. Optimal value is 0 (Möller et al., 2007, Clinton et al., 2010).
- "RPsuper" refers to Relative Position (super) metric. Its values range from 0 (optimal) to 1 (Möller et al., 2007, Clinton et al., 2010).
- "OI2" refers to Overlap Index metric. Its values range from 0 to 1 (optimal) (Yang et al., 2017).

Value

Return a numeric vector with computed metric.

References

A complete list of cited references is available in ?segmetric.

See Also

sm_list_metrics()
Examples

```r
# load sample datasets
data("sample_ref_sf", package = "segmetric")
data("sample_seg_sf", package = "segmetric")

# create segmetric object
m <- sm_read(ref_sf = sample_ref_sf, seg_sf = sample_seg_sf)

# compute AFI metric and summarize it
sm_compute(m, "AFI") %>% summary()

# compute three metrics and summarize them
sm_compute(m, c("AFI", "OS1", "US2")) %>% summary()

# compute OS1, F_measure, and US2 metrics using pipe
m <- sm_compute(m, "OS1") %>%
    sm_compute("F_measure") %>%
    sm_compute("US2")

# summarize them
summary(m)
```

---

**plot**

*Plot function*

**Description**

Plot a segmetric map according to the parameter type:

- "base": simple plot of the reference or segmentation polygons;
- "subset": plot polygons from a subset over the base plot;
- "choropleth": plot a choropleth map from polygons of a subset using metric values.

**Usage**

```r
## S3 method for class 'segmetric'
plot(
  x,
  type = "base",
  ..., 
  title = NULL,
  layers = c("ref_sf", "seg_sf"),
  background = "#FFFFFF",
  ref_color = "#FF00009F",
  ref_fill = "#FFFFFF00",
  ref_label = "reference",
  ref_size = 2,
```
ref_symbol = 2,
seg_color = "#0000009F",
seg_fill = "#FFFFFF00",
seg_label = "segment",
seg_size = 1,
seg_symbol = 3,
selected_fill = "#9A9AFF50",
plot_centroids = TRUE,
centroids_color = "#000000FF",
centroids_label = "centroid",
subset_id = NULL,
subset_color = "#FFFFFF00",
subset_fill = "#F0E4167F",
metric_id = NULL,
break_style = "jenks",
choropleth_palette = "YlGnBu",
choropleth_palette_reverse = FALSE,
choropleth_size = 0.1,
plot_extent = NULL,
plot_legend = TRUE,
plot_axes = TRUE
)

Arguments

x A segmetric object.
type A character. Either "base", "subset", or "choropleth".
... Ignored.
title A character with plot title
layers A character. One or both of "ref_sf" and "seg_sf" (works only for type = "base" and type = "subset").
background A character with valid color used in graph's background.
ref_color, seg_color, ref_fill, seg_fill A character with a valid hexadecimal color in rgb or rgba format. Set the border and fill colors for reference and segmentation polygons.
ref_label, seg_label, centroids_label A character with legend labels for reference polygons, segmentation polygons, and centroids.
ref_size, seg_size A numeric. Set symbol's size for centroids.
ref_symbol, seg_symbol An integer. Symbol to represent polygons' centroids (see pch param in points).
selected_fill A character with a valid hexadecimal color in rgb or rgba format. Set the fill color of selected reference or segmentation polygons depending on subset_id.
plot_centroids A logical. Plot centroids or not.
Description

ref_sf: a dataset containing field boundaries from Luiz Eduardo Magalhaes municipality, Brazil.

The data covers the following extent: xmin: -46.37683 ymin: -12.34579 xmax: -46.15776 ymax: -12.13663 CRS: EPSG:4326

sample_ref_sf: a subset of ref_sf dataset.

Usage

ref_sf

sample_ref_sf

Format

These datasets are objects of class sf (inherited from tbl_df, tbl, data.frame) with 2 variables:

• id: identification
• geometry: polygons

ref_sf: a dataset with 195 features.
sample_ref_sf: a dataset with 5 features.
Source


References


Examples

```r
data("ref_sf", package = "segmetric")
data("sample_ref_sf", package = "segmetric")
```

---

**seg_sf**  
*Segmentation dataset*

---

**Description**

seg200_sf,seg500_sf,seg800_sf,seg1000_sf: a dataset containing segments generated from PlanetScope image, level 3B, acquired on Feb 18, 2020, with 3.7-meter resolution (Planet Team, 2017), using the multiresolution segmentation method (Baatz and Schape, 2000).

The data covers the approximately the same area of LEM+ dataset (see ref_sf).

The data was post-processed using the spectral difference algorithm on band 3.

The polygons were simplified using the Douglas-Peucker algorithm in QGIS.

Self-intersections were removed using SAGA’s Polygon Self-Intersection.

Segmentation parameters:

- scale parameter: 200 (seg200_sf), 500 (seg500_sf), 800 (seg800_sf), and 1000 (seg1000_sf)
- shape: 0.9
- compactness: 0.1

Spectral difference parameters:

- spectral difference: 20

Simplification parameter:

- distance: 10-meters

Only those polygons intersecting reference data with an area-perimeter ratio above 25 were selected.

sample_seg_sf: a subset of seg_sf dataset.
Usage

seg200_sf
seg500_sf
seg800_sf
seg1000_sf
sample_seg_sf

Format

These datasets are objects of class sf (inherited from tbl_df, tbl, data.frame) with 2 variables:

- id: identification
- geometry: polygons


An object of class sf (inherits from tbl_df, tbl, data.frame) with 215 rows and 2 columns.
An object of class sf (inherits from tbl_df, tbl, data.frame) with 169 rows and 2 columns.
An object of class sf (inherits from tbl_df, tbl, data.frame) with 158 rows and 2 columns.
sample_seg_sf: a dataset with 6 features extracted from seg500_sf dataset.

References


Examples

data("seg200_sf", package = "segmetric")
data("seg500_sf", package = "segmetric")
data("seg800_sf", package = "segmetric")
data("seg1000_sf", package = "segmetric")
data("sample_seg_sf", package = "segmetric")
These functions compute subsets required to calculate segmentation metrics as described in Clinton et al. (2010) and Costa et al. (2017).

- `sm_ref()` returns the set of \( n \) polygons of reference, represented by \( X = \{ x_i : i = 1, \ldots, n \} \)
- `sm_seg()` returns the set of \( m \) segmentation polygons, represented by \( Y = \{ y_j : j = 1, \ldots, m \} \)
- `sm_ytilde()` returns \( \tilde{Y}_i \), a subset of \( Y \), where \( \tilde{Y}_i = \{ y_j : \text{area}(x_i \cap y_j) \neq 0 \} \)
- `sm_xtilde()` returns \( \tilde{X}_j \), a subset of \( X \), where \( \tilde{X}_j = \{ x_i : \text{area}(y_j \cap x_i) \neq 0 \} \)
- `sm_yprime()` returns \( Y'_i \), a subset of \( Y \), where \( Y'_i = \{ y_j : \max(\text{area}(x_i \cap y_j)) \} \)
- `sm_xprime()` returns \( X'_j \), a subset of \( X \), where \( X'_j = \{ x_i : \max(\text{area}(y_j \cap x_i)) \} \)
- `sm_ya()` returns \( Y_{a_i} \), a subset of \( \tilde{Y}_i \), where \( Y_{a_i} = \{ y_j : \text{centroid}(x_i) \in y_j \} \)
- `sm_yb()` returns \( Y_{b_i} \), a subset of \( \tilde{Y}_i \), where \( Y_{b_i} = \{ y_j : \text{centroid}(y_j) \in x_i \} \)
- `sm_yc()` returns \( Y_{c_i} \), a subset of \( \tilde{Y}_i \), where \( Y_{c_i} = \{ y_j : \text{area}(x_i \cap y_j) / \text{area}(y_j) > 0.5 \} \)
- `sm_yd()` returns \( Y_{d_i} \), a subset of \( \tilde{Y}_i \), where \( Y_{d_i} = \{ y_j : \text{area}(x_i \cap y_j) / \text{area}(x_i) > 0.5 \} \)
- `sm_ystar()` returns \( Y^*_i \), where \( Y^*_i = Y_{a_i} \cup Y_{b_i} \cup Y_{c_i} \cup Y_{d_i} \)
- `sm_ycd()` returns \( Y_{cd_i} \), where \( Y_{cd_i} = Y_{c_i} \cup Y_{d_i} \)
- `sm_ye()` returns \( Y_{e_i} \), a subset of \( \tilde{Y}_i \), where \( Y_{e_i} = \{ y_j : \text{area}(x_i \cap y_j) / \text{area}(y_j) = 1 \} \)
- `sm_yf()` returns \( Y_{f_i} \), a subset of \( \tilde{Y}_i \), where \( Y_{f_i} = \{ y_j : \text{area}(x_i \cap y_j) / \text{area}(y_j) > 0.55 \} \)
- `sm_yg()` returns \( Y_{g_i} \), a subset of \( \tilde{Y}_i \), where \( Y_{g_i} = \{ y_j : \text{area}(x_i \cap y_j) / \text{area}(y_j) > 0.75 \} \)

**Usage**

- `sm_ytilde(m)`
- `sm_xtilde(m)`
- `sm_yprime(m)`
- `sm_xprime(m)`
- `sm_ya(m)`
- `sm_yb(m)`
- `sm_yc(m)`
- `sm_yd(m)`
subset_handling_functions

sm_ystar(m)
sm_ycd(m)
sm_ye(m)
sm_yf(m)
sm_yg(m)

Arguments

m A segmetric object.

Value

- sm_ref(): Return an object of class ref_sf (inherited from sf) containing identification (ref_id) and geometry (geometry) columns.
- sm_seg(): Return an object of class seg_sf (inherited from sf) containing identification (seg_id) and geometry (geometry) columns.
- sm_ytilde(), sm_xtilde(), sm_yprime(), sm_xprime(), sm_ya(), sm_yb(), sm_yc(), sm_yd(), sm_yestar(), sm_yecd(), sm_ye(), sm_yf(), and sm_yg(): Return an object of class subset_sf (inherited from sf) containing identification (ref_id and seg_id), and geometry (geometry) columns.

References


subset_handling_functions

Subset handling functions

Description

These functions are intended to be used in new metric extensions. They handle subset_sf data (inherited from sf class) stored in segmetric objects.

- sm_list() lists subsets already computed and stored in a segmetric object.
- sm_exists() verifies if a subset_id exists in a segmetric object.
subset_handling_functions

- **sm_subset()** evaluates and stores a subset_sf object.
- **sm_indirect()** finds the subset_id of a given subset_sf object stored in a segmetric object.
- **sm_segmetric()** returns the segmetric object that stores a given subset object (either a ref_sf, a seg_sf, or a subset_sf).
- **sm_get()** retrieves a subset_sf object stored in a segmetric object.
- **sm_inset()** operator equivalent to inner join but returns only objects from s1, or its corresponding row in s2 if parameter return_index is TRUE.
- **sm_group_by()**: Apply a function to groups of subset_sf.

**Usage**

```r
sm_list(m)
sm_exists(m, subset_id)
sm_subset(m, subset_id, expr = NULL)
sm_indirect(s)
sm_segmetric(s)
sm_get(m, subset_id)
sm_ref(m)
sm_seg(m)
sm_inset(s1, s2, return_index = FALSE)
## S3 method for class 'ref_sf'
sm_inset(s1, s2, return_index = FALSE)
## S3 method for class 'seg_sf'
sm_inset(s1, s2, return_index = FALSE)
## S3 method for class 'subset_sf'
sm_inset(s1, s2, return_index = FALSE)
sm_group_by(s, by, fn, ...)
```

**Arguments**

- **m** A segmetric object.
- **subset_id** A character value informing a subset name.
- **expr** A valid piece of code in R inside curly braces. This code is evaluated to generate a subset.
subset_handling_functions

s, s1, s2 Either a ref_sf, a seg_sf, or a subset_sf object.
return_index A logical value indicating if the corresponding rows in s1 should be returned instead of the actual corresponding values of s2.
by A character value with the column to group.
fn A function to apply on each group.
... For sm_group_by(), extra parameter to pass to fn function.

Value

- sm_list(): Return a character vector with all names of subsets stored in the segmetric object.
- sm_exists(): Return a logical value indicating if a given subset name is stored in the segmetric object.
- sm_subset(): Return a subset_sf object.
- sm_indirect(): Return the subset name of a given subset_sf object stored in a segmetric object.
- sm_segmetric(): Return a segmetric object that stores a given subset_sf object.
- sm_get(): Return a subset_sf object stored in a segmetric object.
- sm_inset(): Return either a subset_sf object or an integer vector with the index of corresponding rows of s2 object.
- sm_group_by(): Return a subset_sf object.

Examples

# load sample datasets
data("sample_ref_sf", package = "segmetric")
data("sample_seg_sf", package = "segmetric")

# create segmetric object
m <- sm_read(ref_sf = sample_ref_sf, seg_sf = sample_seg_sf)

# lists only 'ref_sf' and 'seg_sf'
sm_list(m)

# computes 'Y_tilde' subset and stores it as 'test_subset' subset id
# sm_ytilde(m) also stores a subset under 'Y_tilde' id
s <- sm_subset(m, "test_subset", sm_ytilde(m))

# lists 'ref_sf', 'seg_sf', 'test_subset', and 'Y_tilde'
sm_list(m)

# which segmetric object stores 's' subset?
m2 <- sm_segmetric(s)

# m is identical to m2
identical(m, m2)

# which name 's' subset is stored in 'm' segmetric object?
sm_indirect(s)

# retrieve 'test_subset' data from 'm' object
s2 <- sm_get(m, 'test_subset')

# s is identical to s2
identical(s, s2)
Index

* datasets
  * ref_sf, 10
  * seg_sf, 11
  _PACKAGE (segmetric-package), 2
classIntervals, 10
general_functions, 4
metric_functions, 6
plot, 8
points, 9
ref_sf, 10, 11
sample_ref_sf (ref_sf), 10
sample_seg_sf (seg_sf), 11
seg1000_sf (seg_sf), 11
seg200_sf (seg_sf), 11
seg500_sf (seg_sf), 11
seg800_sf (seg_sf), 11
seg_sf, 11
segmetric (segmetric-package), 2
segmetric-package, 2
set_functions, 13
sm_apply_group (general_functions), 4
sm_area (general_functions), 4
sm_centroid (general_functions), 4
sm_compute (metric_functions), 6
sm_distance (general_functions), 4
sm_exists (subset_handling_functions), 14
sm_get (subset_handling_functions), 14
sm_group_by
  (subset_handling_functions), 14
sm_indirect
  (subset_handling_functions), 14
sm_inset (subset_handling_functions), 14
sm_intersection (general_functions), 4
sm_list (subset_handling_functions), 14
sm_metric_subset (metric_functions), 6
sm_norm_left (general_functions), 4
sm_norm_right (general_functions), 4
sm_options (general_functions), 4
sm_rbind (general_functions), 4
sm_ref (subset_handling_functions), 14
sm_seg (subset_handling_functions), 14
sm_segmometric (subset_handling_functions), 14
sm_subset (subset_handling_functions), 14
sm_subset_union (general_functions), 4
sm_summarize_group (general_functions), 4
sm_xprime (set_functions), 13
sm_xtilde (set_functions), 13
sm_ya (set_functions), 13
sm_yb (set_functions), 13
sm yc (set_functions), 13
sm_ycd (set_functions), 13
sm_yd (set_functions), 13
sm_ye (set_functions), 13
sm_yf (set_functions), 13
sm_yg (set_functions), 13
sm_yprime (set_functions), 13
sm_ystar (set_functions), 13
sm_ystar (set_functions), 13
sm_ystar (set_functions), 13
sm_ytilde (set_functions), 13
sm_ytilde (set_functions), 13
subset_handling_functions, 14