

Package ‘divseg’

July 22, 2025

Title Calculate Diversity and Segregation Indices

Version 0.0.5

Date 2022-12-16

Description Implements common measures of diversity and spatial segregation. This package has tools to compute the majority of measures are reviewed in Massey and Denton (1988) <[doi:10.2307/2579183](https://doi.org/10.2307/2579183)>. Multiple common measures of within-geography diversity are implemented as well. All functions operate on data frames with a 'tidyselect' based workflow.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.2

BugReports <https://github.com/christopherkenny/divseg/issues>

URL <https://github.com/christopherkenny/divseg/>,
<https://christophertkenny.com/divseg/>

Suggests roxygen2, testthat (>= 3.0.0)

Imports sf (>= 1.0.0), rlang (>= 0.4.11), dplyr, magrittr, tidyselect,
tibble, units

Depends R (>= 2.10)

Config/testthat/edition 3

NeedsCompilation no

Author Christopher T. Kenny [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-9386-6860>>)

Maintainer Christopher T. Kenny <christopherkenny@fas.harvard.edu>

Repository CRAN

Date/Publication 2022-12-17 08:30:02 UTC

Contents

de_county	2
de_tract	3
ds_abs_cent	3
ds_abs_clust	4
ds_abs_conc	5
ds_atkinson	5
ds_blau	6
ds_correlation	7
ds_dd_interaction	7
ds_dd_isolation	8
ds_delta	9
ds_dissim	10
ds_diversity	10
ds_entropy	11
ds_gini	12
ds_hhi	13
ds_interaction	14
ds_inv_simpson	14
ds_isolation	15
ds_rel_cent	16
ds_rel_clust	16
ds_rel_conc	17
ds_reyni	18
ds_shannon	19
ds_spat_prox	19
Index	21

de_county	<i>de_county</i>
-----------	------------------

Description

This data contains 2010 Census data for each of the three counties in DE.

Usage

```
data("de_county")
```

Format

An sf dataframe with 3 observations

Examples

```
data("de_county")
```

de_tract	<i>de_tract</i>
----------	-----------------

Description

This data contains 2010 Census data for each of the 218 tracts in DE.

Usage

```
data("de_tract")
```

Format

An sf dataframe with 218 observations

Examples

```
data("de_tract")
```

ds_abs_cent	<i>Compute Absolute Centralization</i>
-------------	--

Description

Compute Absolute Centralization

Usage

```
ds_abs_cent(.data, .cols, .name)
```

```
abs_cent(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble with sf geometry
.cols	tidy-select Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
.name	name for column with absolute centralization. Leave missing to return a vector.
...	arguments to forward to ds_abs_cent from abs_cent

Value

a **tibble** or numeric vector if .name missing

Examples

```
data("de_county")
ds_abs_cent(de_county, c(pop_white, starts_with('pop_')))
ds_abs_cent(de_county, c(pop_white, starts_with('pop_')), 'abs_cent')
```

ds_abs_clust	<i>Compute Absolute Clustering</i>
--------------	------------------------------------

Description

Compute Absolute Clustering

Usage

```
ds_abs_clust(.data, .cols, .name)

abs_clust(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble with sf geometry
.cols	tidy-select Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
.name	name for column with absolute clustering. Leave missing to return a vector.
...	arguments to forward to ds_abs_clust from abs_clust

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data("de_county")
ds_abs_clust(de_county, c(pop_white, starts_with('pop_')))
ds_abs_clust(de_county, c(pop_white, starts_with('pop_')), 'abs_clust')
```

ds_abs_conc	<i>Compute Absolute Concentration</i>
-------------	---------------------------------------

Description

Compute Absolute Concentration

Usage

```
ds_abs_conc(.data, .cols, .name)
```

```
abs_conc(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble with sf geometry
.cols	tidy-select Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
.name	name for column with absolute concentration. Leave missing to return a vector.
...	arguments to forward to ds_abs_conc from abs_conc

Value

a **tibble** or numeric vector if .name missing

Examples

```
data("de_county")
ds_abs_conc(de_county, c(pop_black, starts_with('pop_')))
ds_abs_conc(de_county, c(pop_black, starts_with('pop_')), 'abs_conc')
```

ds_atkinson	<i>Compute Atkinson b Index</i>
-------------	---------------------------------

Description

Compute Atkinson b Index

Usage

```
ds_atkinson(.data, .cols, .name, b = 0.5)
```

```
atkinson(..., .data = dplyr::across(everything()))
```

Arguments

.data [tibble](#)

.cols [tidy-select](#) Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.

.name name for column with Atkinson b index. Leave missing to return a vector.

b Default 0.5. Exponent parameter b, where $0 \leq b \leq 1$.

... arguments to forward to ds_atkinson from atkinson

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data('de_county')
ds_atkinson(de_county, c(pop_white, starts_with('pop_')))
ds_atkinson(de_county, starts_with('pop_'), 'atkinson')
```

ds_blau	<i>Compute Blau's Index</i>
---------	-----------------------------

Description

Compute Blau's Index

Usage

```
ds_blau(.data, .cols, .name)

blau(..., .data = dplyr::across(everything()))
```

Arguments

.data [tibble](#)

.cols [tidy-select](#) Columns to compute the measure with.

.name name for column with Blau index. Leave missing to return a vector.

... arguments to forward to ds_blau from blau

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data("de_county")
ds_blau(de_county, starts_with('pop_'))
ds_blau(de_county, starts_with('pop_'), 'blau')
```

ds_correlation	<i>Compute Correlation Index</i>
----------------	----------------------------------

Description

Compute Correlation Index

Usage

```
ds_correlation(.data, .cols, .name)

correlation(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble
.cols	tidy-select Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
.name	name for column with Correlation index. Leave missing to return a vector.
...	arguments to forward to ds_correlation from correlation

Value

a **tibble** or numeric vector if .name missing

Examples

```
data('de_county')
ds_correlation(de_county, c(pop_white, starts_with('pop_')))
ds_correlation(de_county, starts_with('pop_'), 'correlation')
```

ds_dd_interaction	<i>Compute Distance Decay Interaction</i>
-------------------	---

Description

Compute Distance Decay Interaction

Usage

```
ds_dd_interaction(.data, .cols, .name, .comp = FALSE)

dd_interaction(..., .data = dplyr::across(everything()))
```

Arguments

`.data` [tibble](#) with sf geometry

`.cols` [tidy-select](#) Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.

`.name` name for column with distance decay interaction. Leave missing to return a vector.

`.comp` Default is FALSE. FALSE returns the sum, TRUE returns the components.

`...` arguments to forward to `ds_dd_interaction` from `dd_interaction`

Value

a [tibble](#) or numeric vector if `.name` missing

Examples

```
data("de_county")
ds_dd_interaction(de_county, c(pop_black, starts_with('pop_')))
ds_dd_interaction(de_county, c(pop_black, starts_with('pop_')), 'dd_interaction')
```

ds_dd_isolation	<i>Compute Distance Decay Isolation</i>
-----------------	---

Description

Compute Distance Decay Isolation

Usage

```
ds_dd_isolation(.data, .cols, .name, .comp = FALSE)

dd_isolation(..., .data = dplyr::across(everything()))
```

Arguments

`.data` [tibble](#) with sf geometry

`.cols` [tidy-select](#) Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.

`.name` name for column with distance decay isolation. Leave missing to return a vector.

`.comp` Default is FALSE. FALSE returns the sum, TRUE returns the components.

`...` arguments to forward to `ds_dd_isolation` from `dd_isolation`

Value

a [tibble](#) or numeric vector if `.name` missing

Examples

```
data("de_county")
ds_dd_isolation(de_county, c(pop_black, starts_with('pop_')))
ds_dd_isolation(de_county, c(pop_black, starts_with('pop_')), 'dd_isolation')
```

ds_delta

*Compute Delta Index***Description**

Compute Delta Index

Usage

```
ds_delta(.data, .cols, .name, .comp = FALSE)

delta(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble with sf geometry
.cols	tidy-select Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
.name	name for column with delta index. Leave missing to return a vector.
.comp	Default is FALSE. FALSE returns the sum, TRUE returns the components.
...	arguments to forward to ds_delta from delta

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data("de_county")
ds_delta(de_county, c(pop_white, starts_with('pop_')))
ds_delta(de_county, starts_with('pop_'), 'delta')
```

ds_dissim	<i>Compute Dissimilarity Index</i>
-----------	------------------------------------

Description

Compute Dissimilarity Index

Usage

```
ds_dissim(.data, .cols, .name, .comp = FALSE)

dissim(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble
.cols	tidy-select Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
.name	name for column with dissimilarity index. Leave missing to return a vector.
.comp	Default is FALSE. FALSE returns the sum, TRUE returns the components.
...	arguments to forward to ds_dissim from dissim

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data("de_county")
ds_dissim(de_county, c(pop_white, starts_with('pop_')))
ds_dissim(de_county, c(pop_white, starts_with('pop_')), .comp = TRUE)
ds_dissim(de_county, starts_with('pop_'), 'dissim')
```

ds_diversity	<i>Compute Diversity</i>
--------------	--------------------------

Description

This is equivalent to perplexity.

Usage

```
ds_diversity(.data, .cols, .name, q = 1)

diversity(..., .data = dplyr::across(everything()))

ds_perplexity(.data, .cols, .name, q = 1)

perplexity(..., .data = dplyr::across(everything()))
```

Arguments

`.data` [tibble](#)

`.cols` [tidy-select](#) Columns to compute the measure with.

`.name` name for column with diversity. Leave missing to return a vector.

`q` exponent parameter. Default 0. Can not be 1.

`...` arguments to forward to `ds_diversity` from `diversity`

Value

a [tibble](#) or numeric vector if `.name` missing

Examples

```
data('de_county')
ds_diversity(de_county, starts_with('pop_'))
ds_diversity(de_county, starts_with('pop_'), 'diversity')
```

ds_entropy	<i>Compute Entropy Index</i>
------------	------------------------------

Description

Compute Entropy Index

Usage

```
ds_entropy(.data, .cols, .name, .comp = FALSE)

entropy(..., .data = dplyr::across(everything()))
```

Arguments

`.data` [tibble](#)

`.cols` [tidy-select](#) Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.

`.name` name for column with entropy index. Leave missing to return a vector.

`.comp` Default is FALSE. FALSE returns the sum, TRUE returns the components.

`...` arguments to forward to ds_entropy from entropy

Value

a [tibble](#) or numeric vector if `.name` missing

Examples

```
data("de_county")
ds_entropy(de_county, c(pop_white, starts_with('pop_')))
ds_entropy(de_county, starts_with('pop_'), 'entropy')
```

ds_gini

*Compute Gini Index***Description**

Compute Gini Index

Usage

```
ds_gini(.data, .cols, .name, .comp = FALSE)

gini(..., .data = dplyr::across(everything()))
```

Arguments

`.data` [tibble](#)

`.cols` [tidy-select](#) Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.

`.name` name for column with gini index. Leave missing to return a vector.

`.comp` Default is FALSE. FALSE returns the sum, TRUE returns the components.

`...` arguments to forward to ds_gini from gini

Value

a [tibble](#) or numeric vector if `.name` missing

Examples

```
data("de_county")
ds_gini(de_county, c(pop_white, starts_with('pop_')))
ds_gini(de_county, starts_with('pop_'), 'gini')
```

ds_hhi

*Compute Herfindahl-Hirshman Index***Description**

This is equivalent to the Simpson Index.

Usage

```
ds_hhi(.data, .cols, .name)

hhi(..., .data = dplyr::across(everything()))

ds_simpson(.data, .cols, .name)

simpson(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble
.cols	tidy-select Columns to compute the measure with.
.name	name for column with HHI. Leave missing to return a vector.
...	arguments to forward to ds_hhi from hhi

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data("de_county")
ds_hhi(de_county, starts_with('pop_'))
ds_hhi(de_county, starts_with('pop_'), 'blau')
```

ds_interaction	<i>Compute Interaction Index</i>
----------------	----------------------------------

Description

Compute Interaction Index

Usage

```
ds_interaction(.data, .cols, .name, .comp = FALSE)

interaction(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble
.cols	tidy-select Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
.name	name for column with Interaction index. Leave missing to return a vector.
.comp	Default is FALSE. FALSE returns the sum, TRUE returns the components.
...	arguments to forward to ds_interaction from interaction

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data('de_county')
ds_interaction(de_county, c(pop_white, starts_with('pop_')))
ds_interaction(de_county, starts_with('pop_'), 'interaction')
```

ds_inv_simpson	<i>Compute Simpson Index</i>
----------------	------------------------------

Description

Compute Simpson Index

Usage

```
ds_inv_simpson(.data, .cols, .name)

inv_simpson(..., .data = dplyr::across(everything()))
```

Arguments

.data [tibble](#)
 .cols [tidy-select](#) Columns to compute the measure with.
 .name name for column with Simpson Index Leave missing to return a vector.
 ... arguments to forward to ds_inv_simpson from inv_simpson

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data("de_county")
ds_inv_simpson(de_county, starts_with('pop_'))
ds_inv_simpson(de_county, starts_with('pop_'), 'blau')
```

ds_isolation	<i>Compute Isolation Index</i>
--------------	--------------------------------

Description

Compute Isolation Index

Usage

```
ds_isolation(.data, .cols, .name, .comp = FALSE)

isolation(..., .data = dplyr::across(everything()))
```

Arguments

.data [tibble](#)
 .cols [tidy-select](#) Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
 .name name for column with Isolation index. Leave missing to return a vector.
 .comp Default is FALSE. FALSE returns the sum, TRUE returns the components.
 ... arguments to forward to ds_isolation from isolation

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data('de_county')
ds_isolation(de_county, c(pop_white, starts_with('pop_')))
ds_isolation(de_county, starts_with('pop_'), 'isolation')
```

ds_rel_cent	<i>Compute Relative Centralization</i>
-------------	--

Description

Compute Relative Centralization

Usage

```
ds_rel_cent(.data, .cols, .name)

rel_cent(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble with sf geometry
.cols	tidy-select Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
.name	name for column with relative centralization. Leave missing to return a vector.
...	arguments to forward to ds_rel_cent from rel_cent

Value

a **tibble** or numeric vector if .name missing

Examples

```
data("de_county")
ds_rel_cent(de_county, c(pop_white, starts_with('pop_')))
ds_rel_cent(de_county, c(pop_white, starts_with('pop_')), 'rel_cent')
```

ds_rel_clust	<i>Compute Relative Clustering</i>
--------------	------------------------------------

Description

Compute Relative Clustering

Usage

```
ds_rel_clust(.data, .cols, .name)

rel_clust(..., .data = dplyr::across(everything()))
```

Arguments

`.data` [tibble](#) with sf geometry

`.cols` [tidy-select](#) Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.

`.name` name for column with relative clustering. Leave missing to return a vector.

`...` arguments to forward to `ds_rel_clust` from `rel_clust`

Value

a [tibble](#) or numeric vector if `.name` missing

Examples

```
data("de_county")
ds_rel_clust(de_county, c(pop_black, starts_with('pop_')))
ds_rel_clust(de_county, c(pop_black, starts_with('pop_')), 'rel_clust')
```

ds_rel_conc	<i>Compute Relative Concentration</i>
-------------	---------------------------------------

Description

Compute Relative Concentration

Usage

```
ds_rel_conc(.data, .cols, .name)

rel_conc(..., .data = dplyr::across(everything()))
```

Arguments

`.data` [tibble](#) with sf geometry

`.cols` [tidy-select](#) Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.

`.name` name for column with relative concentration. Leave missing to return a vector.

`...` arguments to forward to `ds_rel_conc` from `rel_conc`

Value

a [tibble](#) or numeric vector if `.name` missing

Examples

```
data('de_county')
ds_rel_conc(de_county, c(pop_black, starts_with('pop_')))
ds_rel_conc(de_county, c(pop_black, starts_with('pop_')), 'rel_conc')
```

ds_reyni	<i>Compute Reyni Entropy</i>
----------	------------------------------

Description

Compute Reyni Entropy

Usage

```
ds_reyni(.data, .cols, .name, q = 0)

reyni(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble
.cols	tidy-select Columns to compute the measure with.
.name	name for column with Reyni entropy. Leave missing to return a vector.
q	exponent parameter. Default 0. Can not be 1.
...	arguments to forward to ds_reyni from reyni

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data('de_county')
ds_reyni(de_county, starts_with('pop_'))
ds_reyni(de_county, starts_with('pop_'), 'reyni')
```

ds_shannon	<i>Compute Shannon Index</i>
------------	------------------------------

Description

Compute Shannon Index

Usage

```
ds_shannon(.data, .cols, .name)
```

```
shannon(..., .data = dplyr::across(everything()))
```

Arguments

.data	tibble
.cols	tidy-select Columns to compute the measure with.
.name	name for column with Shannon index. Leave missing to return a vector.
...	arguments to forward to ds_shannon from shannon

Value

a [tibble](#) or numeric vector if .name missing

Examples

```
data("de_county")
ds_shannon(de_county, starts_with('pop_'))
ds_shannon(de_county, starts_with('pop_'), 'shannon')
```

ds_spat_prox	<i>Compute Spatial Proximity</i>
--------------	----------------------------------

Description

Compute Spatial Proximity

Usage

```
ds_spat_prox(.data, .cols, .name)
```

```
spat_prox(..., .data = dplyr::across(everything()))
```

Arguments

<code>.data</code>	<code>tibble</code> with sf geometry
<code>.cols</code>	<code>tidy-select</code> Columns to compute the measure with. Must be at least 2 columns. If more than 2, treats first column as first group and sum of other columns as second.
<code>.name</code>	name for column with spatial proximity. Leave missing to return a vector.
<code>...</code>	arguments to forward to <code>ds_spat_prox</code> from <code>spat_prox</code>

Value

a `tibble` or numeric vector if `.name` missing

Examples

```
data("de_county")
ds_spat_prox(de_county, c(pop_black, starts_with('pop_')))
ds_spat_prox(de_county, c(pop_black, starts_with('pop_')), 'spat_prox')
```

Index

* centralization

ds_abs_cent, 3
ds_rel_cent, 16

* clustering

ds_abs_clust, 4
ds_dd_interaction, 7
ds_dd_isolation, 8
ds_rel_clust, 16
ds_spat_prox, 19

* concentration

ds_abs_conc, 5
ds_delta, 9
ds_rel_conc, 17

* data

de_county, 2
de_tract, 3

* div

ds_blau, 6
ds_diversity, 10
ds_hhi, 13
ds_inv_simpson, 14
ds_reyni, 18
ds_shannon, 19

* evenness

ds_atkinson, 5
ds_dissim, 10
ds_entropy, 11
ds_gini, 12

* exposure

ds_correlation, 7
ds_interaction, 14
ds_isolation, 15

abs_cent (ds_abs_cent), 3
abs_clust (ds_abs_clust), 4
abs_conc (ds_abs_conc), 5
atkinson (ds_atkinson), 5

blau (ds_blau), 6

correlation (ds_correlation), 7

dd_interaction (ds_dd_interaction), 7

dd_isolation (ds_dd_isolation), 8

de_county, 2

de_tract, 3

delta (ds_delta), 9

dissim (ds_dissim), 10

diversity (ds_diversity), 10

ds_abs_cent, 3

ds_abs_clust, 4

ds_abs_conc, 5

ds_atkinson, 5

ds_blau, 6

ds_correlation, 7

ds_dd_interaction, 7

ds_dd_isolation, 8

ds_delta, 9

ds_dissim, 10

ds_diversity, 10

ds_entropy, 11

ds_gini, 12

ds_hhi, 13

ds_interaction, 14

ds_inv_simpson, 14

ds_isolation, 15

ds_perplexity (ds_diversity), 10

ds_rel_cent, 16

ds_rel_clust, 16

ds_rel_conc, 17

ds_reyni, 18

ds_shannon, 19

ds_simpson (ds_hhi), 13

ds_spat_prox, 19

entropy (ds_entropy), 11

gini (ds_gini), 12

hhi (ds_hhi), 13

`interaction (ds_interaction)`, [14](#)
`inv_simpson (ds_inv_simpson)`, [14](#)
`isolation (ds_isolation)`, [15](#)

`perplexity (ds_diversity)`, [10](#)

`rel_cent (ds_rel_cent)`, [16](#)
`rel_clust (ds_rel_clust)`, [16](#)
`rel_conc (ds_rel_conc)`, [17](#)
`reyni (ds_reyni)`, [18](#)

`shannon (ds_shannon)`, [19](#)
`simpson (ds_hhi)`, [13](#)
`spat_prox (ds_spat_prox)`, [19](#)

`tibble`, [3–20](#)