Package: divseg (via r-universe)

July 23, 2024

Title Calculate Diversity and Segregation Indices

Version 0.1.0

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>. 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

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

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, tidyselect, tibble, units

Depends R (>= 4.0.0)

Config/testthat/edition 3

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

RemoteUrl https://github.com/christopherkenny/divseg

RemoteRef HEAD

RemoteSha 3d042a53de7937d6557a4c2af3dc00427f568209

Contents

`de_county` *de_county*

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` *de_tract*

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` *Compute Absolute Centralization*

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` *Compute Absolute Clustering*

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` *Compute Absolute Concentration*

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` *Compute Atkinson b Index*

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 <= b <= 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` *Compute Blau's Index*

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` *Compute Correlation Index*

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` *Compute Distance Decay Interaction*

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` *Compute Distance Decay Isolation*

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` *Compute Dissimilarity Index*

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` *Compute Diversity*

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` *Compute Entropy Index*

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` *Compute Interaction Index*

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` *Compute Simpson Index*

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` *Compute Isolation Index*

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` *Compute Relative Centralization*

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` *Compute Relative Clustering*

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` *Compute Relative Concentration*

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` *Compute Reyni Entropy*

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` *Compute Shannon Index*

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` *Compute Spatial Proximity*

Description

Compute Spatial Proximity

Usage

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

spat_prox(..., .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 spatial proximity. Leave missing to return a vector.
`...`	arguments to forward to ds_spat_prox from spat_prox

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