Package 'tongfen'

October 5, 2026

Type Package

Title Make Data Based on Different Geographies Comparable

Version 0.3.9

Description Several functions to allow comparisons of data across different geographies, in particular for Canadian census data from different censuses.

License MIT + file LICENSE

Encoding UTF-8

ByteCompile yes

LazyData true

NeedsCompilation no

Imports dplyr (>= 1.1.0), tidyr (>= 1.0), sf, tibble, rlang, purrr, stringr, readr, nanoparquet, tools, utils, lifecycle

Suggests knitr, rmarkdown, RColorBrewer, ggplot2, geojsonsf, cancensus, tidycensus, spelling, readxl, scales, microbenchmark, testthat (>= 3.2.0)

VignetteBuilder knitr, rmarkdown

URL https://github.com/mountainMath/tongfen, https://mountainmath.github.io/tongfen/

BugReports https://github.com/mountainMath/tongfen/issues

Language en-US

RdMacros lifecycle

Depends R (>= 4.1)

Config/roxygen2/version 8.1.0

Author Jens von Bergmann [aut, cre] (creator and maintainer)

Maintainer Jens von Bergmann <jens@mountainmath.ca>

Repository CRAN

Date/Publication 2026-10-05 02:20:02 UTC

Contents

add_census_ca_base_variables . . . 2
aggregate_data_with_meta . . . 3
check_tongfen_areas . . . 4
check_tongfen_single_areas . . . 5
estimate_tongfen_correspondence . . . 5
estimate_tongfen_single_correspondence . . . 7
get_correspondence_ca_census_for . . . 8
get_single_correspondence_ca_census_for . . . 8
get_tongfen_ca_census . . . 9
get_tongfen_ca_census_ct_from_da . . . 10
get_tongfen_census_ct . . . 11
get_tongfen_census_da . . . 12
get_tongfen_correspondence_ca_census . . . 13
get_tongfen_correspondence_us_census . . . 14
get_tongfen_us_census . . . 15
meta_for_additive_variables . . . 16
meta_for_ca_census_vectors . . . 17
proportional_reaggregate . . . 18
tongfen_aggregate . . . 19
tongfen_anomaly_joins . . . 20
tongfen_ca_census_ct . . . 22
tongfen_detect_anomalies . . . 23
tongfen_estimate . . . 25
tongfen_estimate_ca_census . . . 26
tongfen_join_correspondence . . . 28
tongfen_join_regions . . . 29
tongfen_tag_largest_overlap . . . 30
vancouver_elections_data_2015 . . . 31
vancouver_elections_data_2019 . . . 31
vancouver_elections_geos_2015 . . . 32
vancouver_elections_geos_2019 . . . 32

Index **33**

`add_census_ca_base_variables` *Generate metadata from Canadian census vectors*

Description

[Maturing]

Add Population, Dwellings, and Household counts to metadata

Usage

```
add_census_ca_base_variables(meta)
```

Arguments

meta	tibble with metadata as for example provided by ‘meta_for_ca_census_vectors‘

Value

tibble with metadata

---

aggregate_data_with_meta

Aggregate variables in grouped data

Description

[Maturing]

Aggregate census data up, assumes data is grouped for aggregation Uses data from meta to determine how to aggregate up

Usage

```
aggregate_data_with_meta(data, meta, geo = FALSE, na.rm = TRUE, quiet = FALSE)
```

Arguments

data	census data as obtained from get_census call, grouped by TongfenID
meta	list with variables and aggregation information as obtained from meta_for_vectors
geo	logical, should also aggregate geographic data
na.rm	logical, should NA values be ignored or carried through.
quiet	logical, don’t emit messages if set to ‘TRUE‘

Value

data frame with variables aggregated to new common geography

Examples

```
# Aggregate population from DA level to grouped by CT_UID
## Not run:
geo <- cancensus::get_census("CA06",regions=list(CSD="5915022"),level='DA')
meta <- meta_for_additive_variables("CA06","Population")
result <- aggregate_data_with_meta(geo %>% group_by(CT_UID),meta)

## End(Not run)
```

check_tongfen_areas *Check geographic integrity*

Description

[Maturing]

Sanity check for areas of estimated tongfen correspondence. This is useful if for example the total extent of geo1 and geo2 differ and there are regions at the edges with large difference in overlap.

The result is a diagnostic, not a pass/fail test. Geographies for different years are simplified independently and differ in how water features are cut out, so a sizable area mismatch does not by itself mean the regions were matched up incorrectly.

Usage

```
check_tongfen_areas(data, correspondence)
```

Arguments

data	a list of geographic data of class sf
correspondence	Correspondence table with columns the unique geographic identifiers for each of the geographies and the TongfenID (and optionally TongfenUID and TongfenMethod) returned by 'estimate_tongfen_correspondence'.

Value

A table with columns 'TongfenID', geo_identifiers, the areas of the aggregated regions corresponding to each geographic identifier column, the tongfen estimation method and the maximum log ratio of the areas.

Examples

```
# Estimate a common geography for 2006 and 2016 dissemination areas in the City of Vancouver
# based on the geographic data and check estimation errors
## Not run:
regions <- list(CSD="5915022")

data_06 <- cancensus::get_census("CA06",regions=regions,geo_format='sf',level="DA") %>%
  rename(GeoUID_06=GeoUID)
data_16 <- cancensus::get_census("CA16",regions=regions,geo_format="sf",level="DA") %>%
  rename(GeoUID_16=GeoUID)

correspondence <- estimate_tongfen_correspondence(list(data_06, data_16),
                                                  c("GeoUID_06","GeoUID_16"))

area_check <- check_tongfen_areas(list(data_06, data_16),correspondence)

## End(Not run)
```

`check_tongfen_single_areas` *Check geographic integrity*

Description

[Deprecated]

Sanity check for areas of estimated tongfen correspondence. This is useful if for example the total extent of geo1 and geo2 differ and there are regions at the edges with large difference in overlap.

Usage

```
check_tongfen_single_areas(geo1, geo2, correspondence)
```

Arguments

`geo1`	input geometry 1 of class sf
`geo2`	input geometry 2 of class sf
`correspondence`	Correspondence table between ‘geo1‘ and ‘geo2‘ as e.g. returned by ‘estimate_tongfen_correspondence‘.

Value

A table with columns ‘TongfenID‘, ‘area1‘ and ‘area2‘, where each row corresponds to a unique ‘TongfenID‘ from them ‘correspondence‘ table and the other columns hold the areas of the regions aggregated from ‘geo1‘ and ‘geo2‘.‘

`estimate_tongfen_correspondence` *Generate tongfen correspondence for list of geographies*

Description

[Maturing]

Get correspondence data for arbitrary congruent geometries. Congruent means that one can obtain a common tiling by aggregating several sub-geometries in each of the two input geo data. Worst case scenario the only common tiling is given by unioning all sub-geometries and there is no finer common tiling.

Usage

```
estimate_tongfen_correspondence(
  data,
  geo_identifiers,
  method = "estimate",
  tolerance = 50,
  computation_crs = NULL
)
```

Arguments

data	list of geometries of class sf
geo_identifiers	vector of unique geographic identifiers for each list entry in data.
method	aggregation method. Possible values are "estimate" or "identifier". "estimate" estimates the correspondence purely from the geographic data. "identifier" assumes that regions with identical geo_identifiers are the same, and uses the "estimate" method for the remaining regions. Default is "estimate".
tolerance	tolerance (in projected coordinate units of 'computation_crs') for feature matching
computation_crs	optional crs in which the computation should be carried out, defaults to crs of the first entry in the data parameter.

Value

A correspondence table linking geo1_uid and geo2_uid with unique TongfenID and TongfenUID columns that enumerate the common geometry.

Examples

```
# Estimate a common geography for 2006 and 2016 dissemination areas in the City of Vancouver
# based on the geographic data.
## Not run:
regions <- list(CSD="5915022")

data_06 <- cancensus::get_census("CA06",regions=regions,geo_format='sf',level="DA") %>%
 rename(GeoUID_06=GeoUID)
data_16 <- cancensus::get_census("CA16",regions=regions,geo_format="sf",level="DA") %>%
  rename(GeoUID_16=GeoUID)

correspondence <- estimate_tongfen_correspondence(list(data_06, data_16),
                                                  c("GeoUID_06","GeoUID_16"))


## End(Not run)
```

`estimate_tongfen_single_correspondence` *Generate tongfen correspondence for two geographies*

Description

[Maturing]

Get correspondence data for arbitrary congruent geometries. Congruent means that one can obtain a common tiling by aggregating several sub-geometries in each of the two input geo data. Worst case scenario the only common tiling is given by unioning all sub-geometries and there is no finer common tiling.

Usage

```
estimate_tongfen_single_correspondence(
  geo1,
  geo2,
  geo1_uid,
  geo2_uid,
  tolerance = 1,
  computation_crs = NULL,
  robust = FALSE
)
```

Arguments

`geo1`	input geometry 1 of class sf
`geo2`	input geometry 2 of class sf
`geo1_uid`	(unique) identifier column for geo1
`geo2_uid`	(unique) identifier column for geo2
`tolerance`	tolerance (in projected coordinate units) for feature matching
`computation_crs`	optional crs in which the computation should be carried out, defaults to crs of geo1
`robust`	boolean parameter, will ensure geometries are valid if set to TRUE

Value

A correspondence table linking geo1_uid and geo2_uid with unique TongfenID and TongfenUID columns that enumerate the common geometry.

`get_correspondence_ca_census_for`
Get StatCan DA or DB level correspondence file

Description

[Deprecated] Joins the StatCan correspondence files for several census years

Usage

```
get_correspondence_ca_census_for(years, level, refresh = FALSE)
```

Arguments

`years`	list of census years
`level`	geographic level, DA or DB
`refresh`	reload the correspondence files, default is 'FALSE'

Value

tibble with correspondence table'spanning all years

`get_single_correspondence_ca_census_for`
Get StatCan DA or DB level correspondence file

Description

[Maturing]

The correspondence files are downloaded from a mirror of the Statistics Canada correspondence files and cached in the tongfen cache directory. The cached files are checked against the mirror once per session and get downloaded again if they changed. The location of the mirror can be changed via the 'tongfen.statcan_correspondence_url' option.

Usage

```
get_single_correspondence_ca_census_for(
  year,
  level = c("DA", "DB"),
  refresh = FALSE
)
```

Arguments

year	census year, only 2006 through 2021 are supported
level	geographic level, DA or DB
refresh	reload the correspondence files, default is 'FALSE'

Value

tibble with correspondence table'

---

get_tongfen_ca_census *Tongfen data from several Canadian censuses*

Description

[Maturing]

Get data from several Canadian censuses on a common geography. Requires sf and cancensus package to be available

Usage

```
get_tongfen_ca_census(
  regions,
  meta,
  level = "CT",
  method = "statcan",
  base_geo = NULL,
  na.rm = FALSE,
  tolerance = 50,
  quiet = FALSE,
  refresh = FALSE,
  crs = NULL,
  data_transform = function(d) d
)
```

Arguments

regions	census region list, should be inclusive list of GeoUIDs across censuses
meta	metadata for the census variables to aggregate, for example as returned by meta_for_ca_census_vectors
level	aggregation level to return data on (default is "CT")
method	tongfen method, options are "statcan" (the default), "estimate", "identifier". * "statcan" method builds up the common geography using Statistics Canada correspondence files, at this point this method only works for "DB", "DA" and "CT" levels. * "estimate" uses 'estimate_tongfen_correspondence' to build up the common geography from scratch based on geographies. * "identifier" assumes regions with identical geographic identifier are identical, and builds up the the correspondence for regions with unmatched geographic identifiers.

base_geo	base census year to build up common geography from, 'NULL' (the default) to not return any geographic data
na.rm	logical, determines how NA values should be treated when aggregating variables, default is 'FALSE'
tolerance	tolerance for 'estimate_tongfen_correspondence' in metres, default value is 50 metres, only used when method is 'estimate' or 'identifier'
quiet	suppress download progress output, default is 'FALSE'
refresh	optional character, refresh data cache for this call, (default 'FALSE')
crs	optional CRS to transform data to, and use for spatial intersections if method is 'identifier' or 'estimate', defaults to '3347' (Statistics Canada Lambert) for the intersections
data_transform	optional transform function to be applied to census data after being returned from cancensus

Value

dataframe with variables on common geography

Examples

```
# Get rent data for census years 2001 through 2016
## Not run:
rent_variables <- c(rent_2001="v_CA01_1667",rent_2016="v_CA16_4901",
                    rent_2011="v_CA11N_2292",rent_2006="v_CA06_2050")
meta <- meta_for_ca_census_vectors(rent_variables)

regions=list(CMA="59933")
rent_data <- get_tongfen_ca_census(regions=regions, meta=meta, quiet=TRUE,
                                   method="estimate", level="CT", base_geo = "CA16")


## End(Not run)
```

---

get_tongfen_ca_census_ct_from_da

Canadian census CT level tongfen via DA correspondence

Description

[Deprecated]

Grab variables from several censuses on a common geography. Requires sf package to be available Will return CT level data

Usage

```
get_tongfen_ca_census_ct_from_da(
  regions,
  vectors,
  geo_format = NA,
  use_cache = TRUE,
  na.rm = TRUE,
  quiet = TRUE
)
```

Arguments

regions	census region list, should be inclusive list of GeoUIDs across censuses
vectors	List of cancensus vectors, can come from different census years
geo_format	'NA' to only get the variables or 'sf' to also get geographic data
use_cache	logical, passed to 'cancensus::get_census' to regulate caching
na.rm	logical, determines how NA values should be treated when aggregating variables
quiet	suppress download progress output, default is 'TRUE'

Value

dataframe with variables on common geography

---

get_tongfen_census_ct *Canadian census CT level tongfen*

Description

[Deprecated]

Grab variables from several censuses on a common geography. Requires sf package to be available Will return CT level data

Usage

```
get_tongfen_census_ct(
  regions,
  vectors,
  geo_format = NA,
  na.rm = TRUE,
  quiet = TRUE,
  refresh = FALSE
)
```

Arguments

regions	census region list, should be inclusive list of GeoUIDs across censuses
vectors	List of cancensus vectors, can come from different census years
geo_format	geographic format for returned data, 'sf' for sf format and 'NA"
na.rm	remove NA values when aggregating up values, default is 'TRUE'
quiet	suppress download progress output, default is 'FALSE'
refresh	optional character, refresh data cache for this call

Value

dataframe with census variables on common geography

---

get_tongfen_census_da *Canadian Census DA level tongfen*

Description

[Deprecated]

Grab variables from several censuses on a common geography. Requires sf package to be available Will return CT level data

Usage

```
get_tongfen_census_da(
  regions,
  vectors,
  geo_format = NA,
  use_cache = TRUE,
  na.rm = TRUE,
  quiet = TRUE
)
```

Arguments

regions	census region list, should be inclusive list of GeoUIDs across censuses
vectors	List of cancensus vectors, can come from different census years
geo_format	'NA' to only get the variables or 'sf' to also get geographic data
use_cache	logical, passed to 'cancensus::get_census' to regulate caching
na.rm	logical, determines how NA values should be treated when aggregating variables
quiet	suppress download progress output, default is 'TRUE'

Value

dataframe with variables on common geography

`get_tongfen_correspondence_ca_census` *Get StatCan correspondence data*

Description

[Maturing]

Get correspondence file for several Canadian censuses on a common geography. Requires sf and cancensus package to be available

Usage

```
get_tongfen_correspondence_ca_census(
  geo_datasets,
  regions,
  level = "CT",
  method = "statcan",
  tolerance = 50,
  quiet = FALSE,
  refresh = FALSE,
  crs = 3347
)
```

Arguments

`geo_datasets`	vector of census geography dataset identifiers
`regions`	census region list, should be inclusive list of GeoUIDs across censuses
`level`	aggregation level to return data on (default is "CT")
`method`	tongfen method, options are "statcan" (the default), "estimate", "identifier". * "statcan" method builds up the common geography using Statistics Canada correspondence files, at this point this method only works for "DB", "DA" and "CT" levels. * "estimate" uses 'estimate_tongfen_correspondence' to build up the common geography from scratch based on geographies. * "identifier" assumes regions with identical geographic identifier are identical, and builds up the the correspondence for regions with unmatched geographic identifiers.
`tolerance`	tolerance for 'estimate_tongfen_correspondence' in metres, default value is 50 metres, only used when method is 'estimate' or 'identifier'
`quiet`	suppress download progress output, default is 'FALSE'
`refresh`	optional character, refresh data cache for this call, (default 'FALSE')
`crs`	CRS to use for the spatial intersections if method is 'identifier' or 'estimate', default is '3347' (Statistics Canada Lambert)

Value

dataframe with the multi-census correspondence file

Examples

```
# Get correspondance files between CTs in 2006 and 2016 censuses in Vancouver CMA
## Not run:
correspondence <- get_tongfen_correspondence_ca_census(geo_datasets=c('CA06','CA16'),
                                                       regions=list(CMA="59933"),level='CT')

## End(Not run)
```

`get_tongfen_correspondence_us_census` *Get correspondence table for US census geographies*

Description

[Maturing]

Builds a correspondence table matching US census geographies across censuses, based on the relationship files published by the US Census Bureau. Censuses that aren't requested but sit in between two that are get traversed on the way, the Census Bureau only publishes relationship files between consecutive censuses.

The relationship files are geometric overlays that list every sliver along boundaries that only shifted slightly. Those get cut via 'min_area_share', keeping them would chain unrelated regions into one common geography.

The correspondence layer reaches back one census further than `get_tongfen_us_census`. The 1990 census is available as 'dec1990' here, but the Census Bureau has retired the 1990 API endpoint, so 1990 data has to be brought in by other means, for example from NHGIS via the ipumsr package, and handed to `tongfen_aggregate` together with this correspondence table.

Usage

```
get_tongfen_correspondence_us_census(
  datasets,
  regions,
  level = "tract",
  min_area_share = 0.01,
  cache_path = getOption("tongfen.cache_path")
)
```

Arguments

`datasets`	vector of censuses to match up, valid values are 'dec1990', 'dec2000', 'dec2010' and 'dec2020' for census tracts, 'dec2000' through 'dec2020' for county subdivisions. At least two censuses are needed.
`regions`	list with regions to query the correspondence for. At this stage, the only valid list is a vector of states, i.e. 'regions = list(state=c("CA","OR"))'
`level`	aggregation level, at this stage the only valid levels are 'tract' and 'county subdivision'.

`min_area_share` minimum share of area two geographies have to have in common to count as related, default is '0.01'. The Census Bureau relationship files list every geometric overlap, lowering this pulls in slivers along boundaries that only shifted slightly and chains unrelated regions into one common geography. Raising it gives finer common geographies at the risk of separating regions that did change. No region is ever dropped, if all of its parts are slivers its largest part is kept.

`cache_path` optional path to cache the relationship files in, defaults to the 'tongfen.cache_path' option. If that is not set the 'tongfen.cache_path' environment variable and the 'custom_data_path' option are used, falling back to a temporary directory

Value

tibble with one row per census geography, a GEOID column for each requested census, and the common geography identified by 'TongfenID' and 'TongfenUID'.

Examples

```
# Match up census tracts for the 1990 and 2000 censuses in Rhode Island
## Not run:
correspondence <- get_tongfen_correspondence_us_census(datasets = c("dec1990","dec2000"),
                                                       regions = list(state="RI"))

## End(Not run)
```

`get_tongfen_us_census` *Get US census data for several censuses on a common geography*

Description

[Maturing]

This wraps data acquisition via the tidycensus package and tongfen on a common geography into a single convenience function.

Data is only available for the 2000, 2010 and 2020 censuses, the Census Bureau has retired the 1990 API endpoint. To tongfen 1990 data, obtain it elsewhere and combine it with a correspondence table from `get_tongfen_correspondence_us_census` via `tongfen_aggregate`.

Usage

```
get_tongfen_us_census(
  regions,
  meta,
  level = "tract",
  survey = "census",
  base_geo = NULL,
  min_area_share = 0.01,
  sumfile = NULL
)
```

Arguments

regions	list with regions to query the data for. At this stage, the only valid list is a vector of states, i.e. ‘regions = list(state=c("CA","OR"))‘
meta	metadata for variables to retrieve
level	aggregation level to return the data on. At this stage, the only valid levels are ’tract’ and ’county subdivision’.
survey	survey to get data for, supported options is "census"
base_geo	dataset to use as base geography, for example ‘"dec2010"‘, has to be one of the datasets in ‘meta‘. Default is ‘NULL‘, which uses the first dataset in ‘meta‘.
min_area_share	minimum share of area two geographies have to have in common to count as related, default is ‘0.01‘, see get_tongfen_correspondence_us_census.
sumfile	summary file to read the variables from, either a single value used for all censuses or a vector named by dataset, for example ‘c(dec2010="sf1", dec2020="dhc")‘. Default is ‘NULL‘, which leaves the choice to tidycensus. Note that tidycensus defaults the 2020 census to the PL 94-171 redistricting file, most 2020 variables need ‘sumfile="dhc"‘.

Value

sf object with (wide form) census variables with census year as suffix (separated by underscore "_").

Examples

```
# Get US census data on population and households for 2000 and 2010 censuses on a uniform geography
# based on census tracts.
## Not run:
variables=c(population="H011001",households="H013001")

meta <- c(2000,2010) %>%
  lapply(function(year){
    v <- variables %>% setNames(paste0(names(.),"_",year))
    meta_for_additive_variables(paste0("dec",year),v)
  }) %>%
  bind_rows()
census_data <- get_tongfen_us_census(regions = list(state="CA"), meta=meta, level="tract") %>%
  mutate(change=population_2010/households_2010-population_2000/households_2000)


## End(Not run)
```

---

meta_for_additive_variables

Generate tongfen metadata for additive variables

Description

[Maturing]

Generates metadata to be used in tongfen_aggregate. Variables need to be additive like counts.

Usage

```
meta_for_additive_variables(dataset, variables)
```

Arguments

dataset	identifier for the dataset containing the variable
variables	(named) vector with additive variables

Value

a tibble to be used in tongfen_aggregate

Examples

```
# Get metadata for additive variable Population for the CA16 and CA06 datasets
## Not run:
meta <- meta_for_additive_variables(c("CA06","CA16"),"Population")

## End(Not run)
```

---

meta_for_ca_census_vectors

Generate metadata from Canadian census vectors

Description

[Maturing]

Build tibble with information on how to aggregate variables given vectors Queries list_census_variables to obtain needed information and add in vectors needed for aggregation

Usage

```
meta_for_ca_census_vectors(vectors)
```

Arguments

vectors	list of variables to query

Value

tidy dataframe with metadata information for requested variables and additional variables needed for tongfen operations

Examples

```
# Build metadata for vectors
## Not run:
meta <- meta_for_ca_census_vectors(c("v_CA16_4836","v_CA16_4838","v_CA16_4899"))

## End(Not run)
```

`proportional_reaggregate` *Dasymetric downsampling*

Description

[Maturing]

Proportionally re-aggregate hierarchical data to lower-level w.r.t. values of the *base* variable Also handles cases where lower level data may be available but blinded at times by filling in data from higher level

Data at lower aggregation levels may not add up to the more accurate aggregate counts. This function distributes the aggregate level counts proportionally (by population) to the containing lower level geographic regions.

Usage

```
proportional_reaggregate(
  data,
  parent_data,
  geo_match,
  categories,
  base = "Population"
)
```

Arguments

`data`	The base geographic data
`parent_data`	Higher level geographic data
`geo_match`	A named string informing on what column names to match data and parent_data
`categories`	Vector of column names to re-aggregate
`base`	Column name to use for proportional weighting when re-aggregating, or named vector with column name for each category. Categories that should be re-aggregated as means should be set to NA and will only be reaggregated if the base data has NA values.

Value

dataframe with downsampled variables from parent_data

Examples

```
# Proportionally reaggregate visible minority data from dissemination area 2016
# census data to dissemination block geography, proportionally based on dissemination
# block population
## Not run:
regions <- list(CSD="5915022")
variables <- cancensus::child_census_vectors("v_CA16_3954")

da_data <- cancensus::get_census("CA16",regions=regions,
                                 vectors=setNames(variables$vector,variables$label),
                                 level="DA")
geo_data <- cancensus::get_census("CA16",regions=regions,geo_format="sf",level="DB")

db_data <- geo_data %>% proportional_reaggregate(da_data,c("DA_UID"="GeoUID"),variables$label)


## End(Not run)
```

`tongfen_aggregate` *Perform tongfen according to correspondence*

Description

[Maturing]

Aggregate variables specified in meta for several datasets according to correspondence.

Usage

```
tongfen_aggregate(
  data,
  correspondence,
  meta = NULL,
  base_geo = NULL,
  na.rm = TRUE
)
```

Arguments

`data`	named list of datasets to be aggregated. The names identify the datasets, they are matched against the 'geo_dataset' column in 'meta' to pick the aggregation rules and labels for each dataset. Without names, or with names not found in 'meta', the rules for all datasets are applied and the variables keep their original names
`correspondence`	correspondence data for gluing up the datasets
`meta`	metadata containing aggregation rules as for example returned by 'meta_for_ca_census_vectors'

base_geo	identifier for which data element to base the final geography on, uses the first data element if 'NULL' (default), expects that 'base_geo' is an element of 'names(data)'.
na.rm	logical, determines how NA values should be treated when aggregating variables, default is 'TRUE'

Value

aggregated dataset of class sf if base_geo is not NULL and data is of type sf or tibble otherwise.

Examples

```
# aggregate census tract level 2006 and 2016 population data on common geography built through
# correspondence from 2006 and 2016 census tracts in the City of Vancouver.
## Not run:
regions <- list(CSD="5915022")
geo1 <- cancensus::get_census("CA06",regions=regions,geo_format='sf',level='CT')
geo2 <- cancensus::get_census("CA16",regions=regions,geo_format='sf',level='CT')
meta <- meta_for_additive_variables(c("CA06","CA16"),"Population")
correspondence <- get_tongfen_correspondence_ca_census(geo_datasets=c('CA06','CA16'),
                                                       regions=regions,level='CT')
result <- tongfen_aggregate(list(CA06=geo1 %>% rename(GeoUIDCA06=GeoUID),
                           CA16=geo2 %>% rename(GeoUIDCA16=GeoUID)),correspondence,meta)

## End(Not run)
```

---

tongfen_anomaly_joins *Determine regions to join to correct for likely geocoding anomalies*

Description

[Experimental]

Looks for regions with surprising drops in the timeline of a count variable that are complemented by a neighbouring region, as explained in 'tongfen_detect_anomalies', and joins them. This gets repeated on the joined regions until there are no more regions left that qualify to get joined. In each round a region only gets joined with one other region, the most surprising regions go first.

Joining regions trades geographic detail for timelines that are consistent over time. The parameters control how aggressively regions get joined and are best calibrated on the data at hand, erring on the side of joining too few regions risks keeping geocoding problems, erring on the other side risks removing real changes and needlessly coarsens the geography.

The result can be used to join the regions via 'tongfen_join_regions', or to update a correspondence via 'tongfen_join_correspondence'.

Usage

```
tongfen_anomaly_joins(
  data,
  variables,
  id = "TongfenID",
  neighbours = NULL,
  rel_scale = 0.25,
  abs_scale = 200,
  p = 4,
  surprise_cutoff = 0.15,
  total_surprise_cutoff = 0.75,
  cutoff_fact = 0.6,
  surprise_reduction_const = 0.15,
  sum_fact = 0.7
)
```

Arguments

data	data on a common geography, with one row per region, for example as returned by ‘tongfen_aggregate‘ or ‘get_tongfen_ca_census‘. Needs to be of class sf unless ‘neighbours‘ is specified.
variables	names of the columns holding the timeline of a count variable like population or dwellings, in temporal order. Changes from or to a missing value are not surprising and don’t make up for surprising changes in neighbouring regions, and joined regions are missing a value if one of the regions they are made up of is. Replace missing values by zero beforehand if they stand for regions where nothing got counted
id	name of the column that uniquely identifies the regions, default is "TongfenID"
neighbours	optional, neighbouring regions as a table with the identifiers of pairs of neighbouring regions in the first two columns, or as a neighbours list like the ones returned by ‘spdep::poly2nb‘. By default all regions with intersecting geometries are neighbours, which can miss neighbours if the geometries have been simplified and don’t share their boundaries any more.
rel_scale	relative decrease that is half way to full surprise, default is ‘0.25‘ for a 25% drop
abs_scale	absolute decrease that is half way to full surprise, default is ‘200‘
p	exponent of the norm used to combine the surprises across the timeline into the total surprise. Large values focus on the most surprising change, 1 adds up the surprises of all changes, default is ‘4‘
surprise_cutoff	changes with larger surprise count as surprising, default is ‘0.15‘. Only regions with at least one surprising change are candidates
total_surprise_cutoff	only regions with larger total surprise are candidates, default is ‘0.75‘
cutoff_fact	join regions if the total surprise after joining is lower than this share of the total surprise of the candidate region, default is ‘0.6‘

surprise_reduction_const
: join regions if joining lowers the total surprise by more than this, default is ‘0.15‘

sum_fact
: join regions if the total surprise after joining is lower than ‘cutoff_fact * sum_fact‘ times the sum of the total surprises of both regions, default is ‘0.7‘

Value

A tibble with one row for each region that gets joined with other regions, with the identifier of the region, the identifier of the joined region it becomes part of in the column named like the identifier with suffix ‘_joined‘, by default ‘TongfenID_joined‘, and the ‘round‘ in which the region first got joined to another region. The identifier of a joined region is the smallest identifier of the regions it is made up of.

Examples

```
# Correct 2001 through 2021 dissemination area level population timelines in the
# City of Vancouver for likely geocoding problems
## Not run:
datasets <- c("CA01","CA06","CA11","CA16","CA21")
meta <- meta_for_additive_variables(datasets,"Population")
data <- get_tongfen_ca_census(regions=list(CSD="5915022"),meta=meta,level="DA",base_geo="CA21")

joins <- tongfen_anomaly_joins(data,paste0("Population_",datasets))
corrected_data <- tongfen_join_regions(data,joins,meta)

## End(Not run)
```

---

tongfen_ca_census_ct *Canadian census CT level tongfen via identifier matching*

Description

[Deprecated]

Aggregate variables to common CTs, returns data2 on new tiling matching data1 geography

Usage

```
tongfen_ca_census_ct(
  data1,
  data2,
  data2_sum_vars,
  data2_group_vars = c(),
  na.rm = TRUE
)
```

Arguments

data1	cancensus CT level dataset for year1 < year2 to serve as base for common geography
data2	cancensus CT level dataset for year2 to be aggregated to common geography
data2_sum_vars	vector of variable names to by summed up when aggregating geographies
data2_group_vars	optional vector of grouping variables
na.rm	optional parameter to remove NA values when summing, default = 'TRUE'

Value

'data2' with the variables in 'data2_sum_vars' aggregated to a common geography matching 'data1', identified by the 'GeoUID' of 'data1'

---

tongfen_detect_anomalies

Detect likely geocoding anomalies in timelines on a common geography

---

Description

[Experimental]

TongFen is only as good as the geocoding that assigned the underlying data to geographic regions in the first place. Geocoding varies over time, and the same dwelling units, and the people living in them, can get assigned to different neighbouring regions in different years. In a timeline on a common geography this shows up as a surprising drop in one region that is offset by a corresponding jump in a neighbouring region.

This function lists the candidate regions with surprising drops in the given count variable, together with the neighbouring region that takes away most of the surprise when both are joined. Use it to check for possible problems and to calibrate the parameters before joining regions with 'tongfen_anomaly_joins'. Not all surprising drops are due to geocoding problems, a drop that is not complemented by a neighbouring region is likely real.

The surprise of a change between two consecutive years ranges from 0 to 1. Only decreases are surprising, the surprise is the product of the surprise of the relative and of the absolute decrease, so that it takes a decrease that is large in both relative and absolute terms to be surprising. The total surprise of a region is the 'p'-norm of the surprises across all changes in the timeline.

A candidate region and its neighbour are flagged for joining if joining reduces the total surprise of the candidate region to below 'cutoff_fact' times its total surprise, or reduces it by more than 'surprise_reduction_const', or reduces it to below 'cutoff_fact * sum_fact' times the sum of the total surprises of both regions. To keep this comparable the surprise after joining is computed from the change of the joined regions relative to the counts of the candidate region alone.

Usage

```
tongfen_detect_anomalies(
  data,
  variables,
  id = "TongfenID",
  neighbours = NULL,
  rel_scale = 0.25,
  abs_scale = 200,
  p = 4,
  surprise_cutoff = 0.15,
  total_surprise_cutoff = 0.75,
  cutoff_fact = 0.6,
  surprise_reduction_const = 0.15,
  sum_fact = 0.7
)
```

Arguments

data	data on a common geography, with one row per region, for example as returned by ‘tongfen_aggregate‘ or ‘get_tongfen_ca_census‘. Needs to be of class sf unless ‘neighbours‘ is specified.
variables	names of the columns holding the timeline of a count variable like population or dwellings, in temporal order. Changes from or to a missing value are not surprising and don’t make up for surprising changes in neighbouring regions, and joined regions are missing a value if one of the regions they are made up of is. Replace missing values by zero beforehand if they stand for regions where nothing got counted
id	name of the column that uniquely identifies the regions, default is "TongfenID"
neighbours	optional, neighbouring regions as a table with the identifiers of pairs of neighbouring regions in the first two columns, or as a neighbours list like the ones returned by ‘spdep::poly2nb‘. By default all regions with intersecting geometries are neighbours, which can miss neighbours if the geometries have been simplified and don’t share their boundaries any more.
rel_scale	relative decrease that is half way to full surprise, default is ‘0.25‘ for a 25% drop
abs_scale	absolute decrease that is half way to full surprise, default is ‘200‘
p	exponent of the norm used to combine the surprises across the timeline into the total surprise. Large values focus on the most surprising change, 1 adds up the surprises of all changes, default is ‘4‘
surprise_cutoff	changes with larger surprise count as surprising, default is ‘0.15‘. Only regions with at least one surprising change are candidates
total_surprise_cutoff	only regions with larger total surprise are candidates, default is ‘0.75‘
cutoff_fact	join regions if the total surprise after joining is lower than this share of the total surprise of the candidate region, default is ‘0.6‘

surprise_reduction_const
: join regions if joining lowers the total surprise by more than this, default is '0.15'

sum_fact
: join regions if the total surprise after joining is lower than 'cutoff_fact * sum_fact' times the sum of the total surprises of both regions, default is '0.7'

Value

A tibble with one row for each candidate region, most surprising first, with the identifier of the region, the number of surprising changes 'surprise_count', the total surprise 'surprise_total', the 'period' with the most surprising change, the identifier of the 'neighbour' that takes away most of the surprise, the total surprise 'surprise_total_joined' after joining both and 'join' indicating if both regions qualify to get joined.

Examples

```
# Check 2001 through 2021 dissemination area level population timelines in the
# City of Vancouver for possible geocoding problems
## Not run:
datasets <- c("CA01","CA06","CA11","CA16","CA21")
meta <- meta_for_additive_variables(datasets,"Population")
data <- get_tongfen_ca_census(regions=list(CSD="5915022"),meta=meta,level="DA",base_geo="CA21")

anomalies <- tongfen_detect_anomalies(data,paste0("Population_",datasets))

## End(Not run)
```

---

tongfen_estimate *Estimate variable values for custom geography*

Description

[Maturing]

Estimates data from source geometry onto target geometry using area-weighted interpolation. The metadata specifies how data should be aggregated, "additive" data like population counts are summed up proportionally to the area of the intersection, "averages" need further additive "parent" count variables to estimate weighted averages.

Usage

```
tongfen_estimate(target, source, meta, na.rm = FALSE)
```

Arguments

target
: custom geography to estimate values for

source
: input geography with values

meta
: metadata for variable aggregation, see 'meta_for_additive_variables' and 'meta_for_ca_census_vectors' for more information on how to construct metadata.

na.rm
: remove NA values when aggregating, default is FALSE

Value

'target' with estimated quantities from 'source' as specified by 'meta', regions in 'target' that don't overlap with 'source' have 'NA' values. Columns in 'target' can't have the same name as the variables to be estimated.

Examples

```
# Estimate 2006 Population in the City of Vancouver dissemination ares on 2016 census geographies
## Not run:
geo1 <- cancensus::get_census("CA06",regions=list(CSD="5915022"),geo_format='sf',level='DA')
geo2 <- cancensus::get_census("CA16",regions=list(CSD="5915022"),geo_format='sf',level='DA')
meta <- meta_for_additive_variables("CA06","Population")
result <- tongfen_estimate(geo2 %>% rename(Population_2016=Population),geo1,meta)

## End(Not run)
```

---

tongfen_estimate_ca_census

Tongfen estimate data for given geometry

Description

[Maturing]

Estimates values for the given census vectors for the given geometry using data from the specified level range. This is a wrapper around 'cancensus::get_intersecting_geometries' and 'tongfen_estimate', optionally with downsampling via 'proportional_reaggregate', to streamline estimating Canadian census data on custom geographies.

Usage

```
tongfen_estimate_ca_census(
  geometry,
  meta,
  level,
  intersection_level = level,
  downsample_level = NULL,
  na.rm = FALSE,
  quiet = FALSE
)
```

Arguments

geometry	geometry
meta	metadata for the census variables to aggregate, for example as returned by 'meta_for_ca_census_vectors'. At this point this function only accepts variables from the same census geography year. We will expand this to also allow estimates across multiple census geography years, but this requires further attention to detail. It is recommended

to apply due caution when running this function separately across several census geography years with the purpose of comparing data across time as a naive application can lead to systematic biases.

`level` level to use for tongfen

`intersection_level`
level to use for geometry intersection, if different from tongfen level by `meta_for_ca_census_vectors`. This can be set at a higher aggregation level to conserve API points for the 'get_intersecting_geometries' call.

`downsample_level`
default 'NULL', can be a geographic level lower than 'level', in which case the data is downsamples to that geography level proportionally using the value of the 'downsample' column (must be supplied) in the 'meta' argument before intersecting the geometries. This can lead to more accurate results. At this point the only allowed variables for the 'downsample' column in 'meta' are "Population", "Households" or "Dwellings", and it can only be one of these for all variables.

`na.rm` how to deal with NA values, default is `FALSE`.

`quiet` suppress progress messages

Value

'geometry' with the estimated values for the census variables specified by 'meta'

Examples

```
# Estimate the 2016 population within 1 km of Toronto City Hall from dissemination area level
# census data
## Not run:
toronto_city_hall <- sf::st_point(c(-79.3839,43.6534)) %>%
  sf::st_sfc(crs=4326) %>%
  sf::st_transform(3348) %>%
  sf::st_buffer(1000) %>%
  sf::st_sf()

meta <- meta_for_additive_variables("CA16","Population")

data <- tongfen_estimate_ca_census(toronto_city_hall,meta,level="DA",intersection_level="CT")

print(paste0("Approximately ",scales::comma(data$Population,accuracy=100),
             " people live within a 1 km radius of Toronto City Hall."))


## End(Not run)
```

tongfen_join_correspondence *Join regions in a correspondence*

Description

[Experimental]

Updates a correspondence so that the given regions are joined, for example to correct for likely geocoding anomalies as determined by ‘tongfen_anomaly_joins‘. The updated correspondence can be used in ‘tongfen_aggregate‘ to aggregate data on the coarser common geography, which works for all variables ‘tongfen_aggregate‘ can deal with and for data that was not part of detecting the anomalies.

Usage

```
tongfen_join_correspondence(correspondence, joins)
```

Arguments

correspondence	correspondence table with columns the unique geographic identifiers for each of the geographies and the TongfenID and TongfenUID, as for example returned by ‘estimate_tongfen_correspondence‘ or ‘get_tongfen_correspondence_ca_census‘
joins	table with the regions to join as returned by ‘tongfen_anomaly_joins‘, with columns ‘TongfenID‘ and ‘TongfenID_joined‘

Value

The correspondence with updated TongfenID and TongfenUID for the regions that got joined. If the correspondence has a TongfenMethod column "anomaly" gets added to the method of the regions that got joined.

Examples

```
# Correct for likely geocoding problems in dissemination area level population timelines
# and use the updated correspondence to aggregate data on the corrected common geography
## Not run:
regions <- list(CSD="5915022")
datasets <- c("CA01","CA06","CA11","CA16","CA21")
meta <- meta_for_additive_variables(datasets,"Population")
data <- get_tongfen_ca_census(regions=regions,meta=meta,level="DA",base_geo="CA21")
joins <- tongfen_anomaly_joins(data,paste0("Population_",datasets))

correspondence <- get_tongfen_correspondence_ca_census(geo_datasets=datasets,
                                                       regions=regions,level="DA") %>%
  tongfen_join_correspondence(joins)

## End(Not run)
```

`tongfen_join_regions` *Join regions in data on a common geography*

Description

[Experimental]

Joins regions in data that has already been aggregated to a common geography, for example to correct for likely geocoding anomalies as determined by ‘tongfen_anomaly_joins‘. The data, and the geometries if the data is of class sf, of the regions that get joined are aggregated, all other regions are left as they are.

Variables are aggregated according to the metadata, numeric variables that are not part of the metadata are assumed to be additive. Variables that are not additive, like averages, can only be aggregated if their parent variable is part of the data. If that is not the case use ‘tongfen_join_correspondence‘ to update the correspondence the data was built from and aggregate the original data again with ‘tongfen_aggregate‘.

Usage

```
tongfen_join_regions(data, joins, meta = NULL, id = "TongfenID", na.rm = TRUE)
```

Arguments

`data`	data on a common geography, with one row per region, for example as returned by ‘tongfen_aggregate‘ or ‘get_tongfen_ca_census‘
`joins`	table with the regions to join as returned by ‘tongfen_anomaly_joins‘, with the identifier of the region and the identifier of the joined region it becomes part of in the column named like the identifier with suffix ‘_joined‘
`meta`	optional metadata containing aggregation rules as for example returned by ‘meta_for_ca_census_vectors‘, variables are matched by their label. Numeric variables that are not part of the metadata are treated as additive, if ‘NULL‘ (the default) that is the case for all numeric variables
`id`	name of the column that uniquely identifies the regions, default is "TongfenID"
`na.rm`	logical, determines how NA values should be treated when aggregating variables, default is ‘TRUE‘

Value

The data with the regions joined. Joined regions take the place and the identifier of the region with the smallest identifier among the regions they are made up of. Variables that are not numeric and not part of the metadata are ‘NA‘ for joined regions.

Examples

```
# Correct 2001 through 2021 dissemination area level population timelines in the
# City of Vancouver for likely geocoding problems
## Not run:
datasets <- c("CA01","CA06","CA11","CA16","CA21")
meta <- meta_for_additive_variables(datasets,"Population")
data <- get_tongfen_ca_census(regions=list(CSD="5915022"),meta=meta,level="DA",base_geo="CA21")

joins <- tongfen_anomaly_joins(data,paste0("Population_",datasets))
corrected_data <- tongfen_join_regions(data,joins,meta)

## End(Not run)
```

---

tongfen_tag_largest_overlap

Tag regions by largest overlap

Description

[Maturing]

tags regions in ‘source‘ by ‘target_id‘ of region in ‘target‘ with the largest overlap

Usage

```
tongfen_tag_largest_overlap(source, target, target_id)
```

Arguments

source	input geography
target	custom geography
target_id	name of the column in ‘target‘ table with unique id (character)

Value

‘source‘ with extra column with the name given by ‘target_id‘ and column ‘...overlap_fraction‘ with the proportion of the area of the source region that overlaps with the region in ‘target‘ with that id

Examples

```
# Tag 2016 dissemination areas in the City of Vancouver by the 2006 census tract they overlap
# the most with
## Not run:
geo1 <- cancensus::get_census("CA06",regions=list(CSD="5915022"),geo_format='sf',level='CT')
geo2 <- cancensus::get_census("CA16",regions=list(CSD="5915022"),geo_format='sf',level='DA')
result <- tongfen_tag_largest_overlap(geo2,geo1 %>% select(CT_2006=GeoUID),"CT_2006")

## End(Not run)
```

vancouver_elections_data_2015 *A dataset with polling station votes data from the 2015 federal election in the Vancouver area*

Description

A dataset with polling station votes data from the 2015 federal election in the Vancouver area

Author(s)

Elections Canada

References

https://www.elections.ca/content.aspx?section=res&dir=rep/off&document=index&lang=e#42GE

vancouver_elections_data_2019 *A dataset with polling station votes data from the 2019 federal election in the Vancouver area*

Description

A dataset with polling station votes data from the 2019 federal election in the Vancouver area

Author(s)

Elections Canada

References

https://www.elections.ca/content.aspx?section=res&dir=rep/off&document=index&lang=e#43GE

vancouver_elections_geos_2015 *A dataset with polling district geographies from the 2015 federal election in the Vancouver area*

Description

A dataset with polling district geographies from the 2015 federal election in the Vancouver area

Author(s)

Elections Canada

References

https://www.elections.ca/content.aspx?section=res&dir=rep/off&document=index&lang=e#42GE

vancouver_elections_geos_2019 *A dataset with polling district geographies from the 2019 federal election in the Vancouver area*

Description

A dataset with polling district geographies from the 2019 federal election in the Vancouver area

Author(s)

Elections Canada

References

https://www.elections.ca/content.aspx?section=res&dir=rep/off&document=index&lang=e#43GE

Index

∗ **base**
  proportional_reaggregate, 18
∗ **data**
  vancouver_elections_data_2015, 31
  vancouver_elections_data_2019, 31
  vancouver_elections_geos_2015, 32
  vancouver_elections_geos_2019, 32
∗ **proportionally**
  proportional_reaggregate, 18
∗ **reaggregate**
  proportional_reaggregate, 18
∗ **variable**
  proportional_reaggregate, 18
∗ **wrt**
  proportional_reaggregate, 18

add_census_ca_base_variables, 2
aggregate_data_with_meta, 3

check_tongfen_areas, 4
check_tongfen_single_areas, 5

estimate_tongfen_correspondence, 5
estimate_tongfen_single_correspondence, 7

get_correspondence_ca_census_for, 8
get_single_correspondence_ca_census_for, 8
get_tongfen_ca_census, 9
get_tongfen_ca_census_ct_from_da, 10
get_tongfen_census_ct, 11
get_tongfen_census_da, 12
get_tongfen_correspondence_ca_census, 13
get_tongfen_correspondence_us_census, 14, *15, 16*
get_tongfen_us_census, *14*, 15

meta_for_additive_variables, 16
meta_for_ca_census_vectors, 17
proportional_reaggregate, 18

tongfen_aggregate, *14, 15*, 19
tongfen_anomaly_joins, 20
tongfen_ca_census_ct, 22
tongfen_detect_anomalies, 23
tongfen_estimate, 25
tongfen_estimate_ca_census, 26
tongfen_join_correspondence, 28
tongfen_join_regions, 29
tongfen_tag_largest_overlap, 30

vancouver_elections_data_2015, 31
vancouver_elections_data_2019, 31
vancouver_elections_geos_2015, 32
vancouver_elections_geos_2019, 32