Package: naomi.utils (via r-universe)

August 13, 2024

Title Utility Functions For Naomi Datasets

Version 0.0.13

Description This package contains utility functions for creating and manipulating datasets for the Naomi model and related projects.

License MIT + file LICENSE

URL https://github.com/mrc-ide/naomi.utils

BugReports https://github.com/mrc-ide/naomi.utils/issues

Additional_repositories https://mrc-ide.r-universe.dev

Imports dplyr, exactextractr, forcats, ggplot2, haven, hintr, lubridate, magrittr, raster, rdhs (>= 0.7.1), sf, spud, survey, tidyr

Suggests testthat

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Repository https://mrc-ide.r-universe.dev

RemoteUrl https://github.com/mrc-ide/naomi.utils

RemoteRef master

RemoteSha 855662cde15847aa42619327f235ddb8751e4774

Contents

`allocate_areas_survey_regions`

Allocate areas to survey regions

Description

Allocate areas at the most granular level to survey regions via spatial join based on largest overlapping area.

Usage

```
allocate_areas_survey_regions(areas_wide, survey_region_boundaries)
```

Arguments

`areas_wide`	wide format area hierarchy, created by `naomi::spread_areas()`.
`survey_region_boundaries`	survey_region_boundaries dataset created by `create_survey_boundaries_dhs()`.

Details

The function `sf:st_join(..., largest = TRUE)` is used to construct a spatial join based on the area of largest overlap.

If the mapping is clean, the following should be satisfied:

1. All areas are allocated to a survey region. This might not happen if an area is non-overlapping with the survey geometry.
2. All survey regions should contain some areas. This might not happen if all areas overlapping a region are not cleanly nested and have a larger overlap with other regions.

The function `assert_survey_region_areas()` implements these checks.

These conditions are not comprehensive and do not guarantee the mapping is accurate, but will catch some basic errors.

Value

A simple features data frame consisting of a mapping of all areas to a survey_region_id.

`assert_area_id_check` *Checks for consistent area IDs between two datasets*

Description

Checks for consistent area IDs between two datasets

Usage

```
assert_area_id_check(df1, df2, key)
```

Arguments

`df1`	a dataframe containing area_id
`df2`	a dataframe containing area_id
`key`	list of columns to compare

Value

If unique area IDs are present between the two datasets, an error will be generated along with a map of mismatching IDs

assert_pop_age_group *Checks valid age groups*

Description

Checks valid age groups

Usage

```
assert_pop_age_group(var)
```

Arguments

var	a value in dataframe extracted using $ notation

Value

If additional age groups are present or missing values for age group, an error will be generated

assert_pop_data_check *Data validation for input population data*

Description

Checks for: (1) consistent aread IDs between pop data and boundaries file (2) pop data age groups consistent with naomi age groups

Usage

```
assert_pop_data_check(pop_data, boundaries)
```

Arguments

pop_data	population dataframe
boundaries	boundary file conatining area IDs

Value

If unique area IDs are present between the two datasets, an error will be generated along with a map of mismatching IDs. If unique age groups area present, an error will be generated.

`assign_dhs_cluster_areas` *Assign survey clusters to dataset areas*

Description

Assign each survey cluster with geocoordinates to an area, ensuring that the assigned area is contained in the specified survey region.

Usage

```
assign_dhs_cluster_areas(survey_clusters, survey_region_areas)
```

Arguments

`survey_clusters`
: Interim survey clusters dataset created by `create_survey_clusters_dhs()`.

`survey_region_areas`
: Dataset of the areas contained in each survey region, created by `allocate_areas_survey_region`
survey_region_areas is a list of candidate location areas for each cluster. Join candidate areas and then select the nearest area based on distance. Usually the coordinate should be contained (distance = 0)

Details

For each survey cluster with geographic coordinates, the area ID containing the cluster is assigned by:

1. Identify all areas contained in the survey region in which the cluster is located. This comprises the set of candidate areas where could be located.
2. Calculate the nearest distance from the cluster coordinates to each candidate area. This distance is 0 if a cluster is contained in an area.
3. Select the area as the area with the nearest distance, in most cases an area containing the cluster (distance = 0).

sf::st_distance() is substantially slower than sf::st_join(). This function could be (maybe much) more efficient by first using st_join() to assign the majority of clusters that are contained in an area, then calculating the distance for the remaining clusters that were not contained inside any of the candidate areas.

Value

Survey clusters dataset with an area_id assigned for each cluster with geographic coordinates and formatted conforming to survey_clusters schema.

`calc_survey_hiv_indicators` *Calculate age/sex/area stratified survey estimates for biomarker outcomes*

Description

Calculate age/sex/area stratified survey estimates for biomarker outcomes

Usage

```
calc_survey_hiv_indicators(
  survey_meta,
  survey_regions,
  survey_clusters,
  survey_individuals,
  survey_biomarker,
  areas,
  sex = c("male", "female", "both"),
  age_group_include = NULL,
  area_top_level = min(areas$area_level),
  area_bottom_level = max(areas$area_level),
  artcov_definition = c("both", "arv", "artself"),
  by_res_type = FALSE
)
```

Arguments

`survey_meta`	Survey metadata.
`survey_regions`	Survey regions.
`survey_clusters`	Survey clusters.
`survey_individuals`	Survey individuals.
`survey_biomarker`	Survey biomarkers.
`areas`	Areas.
`sex`	Sex.
`age_group_include`	Vector of age agroups to include
`area_top_level`	Area top level.
`area_bottom_level`	Area bottom level.
`artcov_definition`	Definition to use for calculate ART coverage.
`by_res_type`	Whether to stratify estimates by urban/rural res_type; logical.

Details

All other data will be subsetted based on the `survey_id` values appearing in survey_meta, so if only want to calculate for a subset of surveys it is sufficient to pass subset for survey_meta and full data frames for the others.

Much of this function needs to be parsed out into more generic functions and rewritten to be more efficient.

- Age group would be more efficient if traversing a tree structure.
- Need generic function to calculate
- Flexibility about age/sex stratifications to calculate.

The argument `artcov_definition` controls whether to use both ARV biomarker and self-report (`artcov_definition = "both"`; default), ARV biomarker only (`artcov_definition = "arv"`), or self-report ART use only (`artcov_definition = "artself"`). If option is `"both"`, then all HIV positive are used as the denomiator and no missing data on either indicator are incorporated. If the option is `"arv"` or `"artself"` then missing values in those variables, respectively, are treated as missing.

`check_boundaries` *Check full and aggregated boundaries*

Description

This function is useful for checking level of coarseness of a simplified versus raw shapefile and any slivers in a shapefile.

Usage

```
check_boundaries(sh1, sh2 = NULL)
```

Arguments

`sh1`	Bottom shapefile with red boundaries
`sh2`	Top shapefile with red boundaries

check_pjnz_shiny90 *Check whether PJNZ contains .shiny90 file*

Description

Check whether PJNZ contains .shiny90 file

Usage

```
check_pjnz_shiny90(pjnz)
```

Arguments

pjnz	file path to PJNZ

Details

TODO: Check whether the .shiny90 file is valid.

Value

Logical whether PJNZ file contains a .shiny90 file

compare_boundaries *Compare boundaries of two shapefiles by overlaying them*

Description

Compare boundaries of two shapefiles by overlaying them

Usage

```
compare_boundaries(sh1, sh2 = NULL, aggregate = FALSE)
```

Arguments

sh1	is bottom shapefile with red boundaries
sh2	is top shapefile with red boundaries
aggregate	whether to aggregate shapefiles

copy_pjnz_extract *Extract the .DP and .PJN from a Spectrum PJNZ*

Description

Copy a PJNZ file to a new location an delete everything except for the .DP and .PJN files.

Usage

```
copy_pjnz_extract(pjnz, out, shiny90 = NULL, force_shiny90 = FALSE)
```

Arguments

pjnz	file path to source PJNZ
out	file path to save output
shiny90	file path to external .shiny90 zip (optional)
force_shiny90	Logical whether or not to force replacement of a .shiny90 file already in the PJNZ with the provided path. The default behaviour is not to replace the .shiny90 file if it already exists in the PJNZ.

Details

Both pjnz and out must be length 1. To apply to multiple files, use `Map` function, e.g. `Map(copy_pjnz_extract, pjnz_list, out_list)`.

The file must be renamed (pjnz cannot equal out) to avoid inadvertently deleting components from an archived PJNZ file.

The default 'force_shiny90 = FLASE)

create_individual_hiv_dhs *Create individual HIV outcomes dataset from DHS*

Description

Create dataset of indiviaul demographic and HIV outcomes.

Usage

```
create_individual_hiv_dhs(surveys, clear_rdhs_cache = FALSE)
```

Arguments

surveys	data.frame of surveys, returned by `create_surveys_dhs()`.

Details

The following fields are extracted:

- survey_id
- cluster_id
- household
- line
- sex
- age
- dob_cmc
- interview_cmc
- indweight
- hivstatus
- arv
- artself
- vls
- cd4
- artall
- hivweight

Value

data.frame consisting of survey ID, cluster ID and individual demographic and HIV outcomes. See details.

Examples

```
## Not run:
surveys <- create_surveys_dhs("MWI")
individuals <- create_individual_hiv_dhs(surveys)

## End(Not run)
```

`create_surveys_dhs` *Create surveys dataset from DHS API*

Description

Construct a surveys dataset from DHS API. Usess `rdhs` to identify the DHS country code from the ISO3, selects relevant surveys, then constructs the `survey_id` and `survey_mid_calendar_quarter`.

Usage

```
create_surveys_dhs(
  iso3,
  survey_type = c("DHS", "AIS", "MIS"),
  survey_characteristics = 23
)
```

Arguments

`iso3`	Three letter ISO3 country code.
`survey_type`	DHS survey types to access. See `?rdhs::dhs_surveys`.
`survey_characteristics`	DHS survey characteristic IDs to filter on See `?rdhs::dhs_survey_characteristics`.

Value

A data frame containing the response from the *dhs_surveys* API endpoint and the `survey_id` and `survey_mid_calendar_quarter`.

Examples

```
## Not run:
create_surveys_dhs("MWI")

## End(Not run)
```

`create_survey_boundaries_dhs` *Create survey region boundaries dataset from DHS spatial data repository*

Description

Create survey region boundaries dataset from DHS spatial data repository

Usage

```
create_survey_boundaries_dhs(
  surveys,
  levelrnk_select = NULL,
  verbose_download = FALSE
)
```

Arguments

`surveys`	data.frame of surveys, returned by `create_surveys_dhs()`.
`levelrnk_select`	A named vector specifying which LEVELRNK to select for a given survey if multiple level ranks are available. Defaults to NULL in which the level with the largest number of regions is selecteed. See details.
`verbose_download`	Whether to print messages from `rdhs::download_boundaries()`. Default is `FALSE`.

Details

For some surveys, the DHS spatial data repository and the survey clusters datasets boundaries at multiple levels (e.g. admin 1 and admin 2). In these cases, the admin level with the largest number or regions is selected by default. The options for multiple level surveys will be printed as messages. To selected a different level supply a named vector with `survey_id` / `LEVELRNK` pairs, for example `levelrnk_select = c("MWI2015DHS" = 1)`. See examples.

Value

A simple features data frame containing DHS region code, region name, and region boundaries for each survey.

Examples

```
## Not run:
surveys <- create_surveys_dhs("MWI")

region_boundaries <- create_survey_boundaries_dhs(surveys)

## Select three regions
levelrnk_select = c("MWI2015DHS" = 1)
region_boundaries <- create_survey_boundaries_dhs(surveys, levelrnk_select)

## End(Not run)
```

---

create_survey_circumcision_dhs

Create male circumcision outcomes dataset from DHS

Description

Create male circumcision outcomes dataset from DHS

Usage

```
create_survey_circumcision_dhs(surveys, clear_rdhs_cache = FALSE)
```

Arguments

`surveys`	data.frame of surveys, returned by `create_surveys_dhs()`.

Details

The following fields are extracted:

- survey_id
- individual_id
- circumcised
- circ_age
- circ_where
- circ_who

Value

data.frame consisting of survey ID, individual ID and male circumcision outcomes. See details.

Examples

```
## Not run:
surveys <- create_surveys_dhs("MWI")
circ <- create_circumcision_dhs(surveys)

## End(Not run)
```

`create_survey_clusters_dhs` *Create survey clusters dataset*

Description

Create survey clusters dataset from DHS household recode and geocluster datasets.

Usage

```
create_survey_clusters_dhs(surveys, clear_rdhs_cache = FALSE)
```

Arguments

`surveys`	data.frame of surveys, returned by `create_surveys_dhs()`.

Value

data.frame consisting of survey clusters, survey region id, and cluster geographic coordinates if available.

Examples

```
## Not run:
surveys <- create_surveys_dhs("MWI")
survey_regions <- create_survey_boundaries_dhs(surveys)
surveys <- surveys_add_dhs_regvar(surveys, survey_regions)

survey_clusters <- create_survey_clusters_dhs(surveys)

## End(Not run)
```

---

create_survey_individuals_dhs

Create survey individuals and biomarker dataset from DHS extract

---

Description

Create survey individuals and biomarker dataset from DHS extract

Usage

```
create_survey_individuals_dhs(dat)

create_survey_biomarker_dhs(dat)
```

Arguments

`dat`	data.frame of merged individual extract, returned by `create_individual_hiv_dhs()`.

Value

data.frame matching UNAIDS data schema

---

create_survey_meta_dhs

Create DHS survey meta data table

Description

Create DHS survey meta data table

Usage

```
create_survey_meta_dhs(surveys)
```

Arguments

`surveys`	data.frame of surveys, returned by `create_surveys_dhs()`.

Value

data.frame of survey metadata specification.

Examples

```
## Not run:
surveys <- create_surveys_dhs("MWI")
survey_meta <- create_survey_meta_dhs(surveys)

## End(Not run)
```

`create_survey_regions_dhs` *Create survey regions dataset from DHS*

Description

Construct survey regions dataset by identifying the smallest area_id that contains the whole survey region.

Usage

```
create_survey_regions_dhs(survey_region_areas)
```

Arguments

`survey_region_areas`	Area allocation to survey regions, created by `allocate_areas_survey_regions()`

Value

Survey regions dataset conforming to schema.

gather_areas *Convert nested hierarchy from wide to long format*

Description

Convert nested hierarchy from wide to long format

Usage

```
gather_areas(x)
```

Arguments

x	Wide format nested hierarchy.

generate_area_id *Generate single Naomi area id*

Description

Generate a Naomi area ID consisting of ISO3, area level and a random `nchar` digit alpha numeric.

Usage

```
generate_area_id(iso3, level, nchar = 5)
```

Arguments

iso3	three character ISO3 code
level	area level as an integer
nchar	number of alpha numeric digits to generate

Details

This function is not vectorized. It generates a single area ID.

This function does not set the seed. Ensure to set the seed before calling the function if you want to reproduce the same results.

Value

An area_id in the format `<ISO3>_<level>_<xyz12>`.

Examples

```
generate_area_id("ISO", 1)
```

get_mid_calendar_quarter *Find Calendar Quarter Midpoint of Two Dates*

Description

Find Calendar Quarter Midpoint of Two Dates

Usage

```
get_mid_calendar_quarter(start_date, end_date)
```

Arguments

start_date	vector coercibel to Date
end_date	vector coercibel to Date

Value

A vector of calendar quarters

Examples

```
start <- c("2005-04-01", "2010-12-01", "2016-01-01")
end <-c("2005-08-01", "2011-05-01", "2016-06-01")

mid_calendar_quarter <- get_mid_calendar_quarter(start, end)
```

hintr_inputs_ready *Prepare output from hintr debug rds for debugging*

Description

Prepare output from hintr debug rds for debugging

Usage

```
hintr_inputs_ready(jobid, root = ".")
```

Arguments

jobid	The issue ID, the name of the folder in sharepoint
root	The debug root dir

Value

Path to local debug

naomi_debug *Download debug from server and upload into sharepoint*

Description

Download debug from server and upload into sharepoint

Usage

```
naomi_debug(
  id,
  jobid,
  dest_folder = "Shared Documents/2023_debug",
  server = NULL
)
```

Arguments

id	The model fit or calibrate ID to download debug for
jobid	The issue ID, the name of the folder to create in sharepoint
dest_folder	The root destination folder in sharepoint
server	The folder to download debug from, defaults to production server

Value

Path to local debug

naomi_extract_gpw *Extract Gridded Population of the World (GPW) raster data*

Description

Extract Gridded Population of the World (GPW) raster data

Usage

```
naomi_extract_gpw(areas, gpw_path = "~/Data/population/GPW 4.11/")
```

Arguments

areas	Naomi area hierarchy dataset with boundaries.
gpw_path	Local path to GPW v4.11 raster files.

Details

This function relies on accessing GPW population files via a local path to the GPW v4.11 rasters because the files are very large.

Datasets are downloaded from:

- Age/sex stratified populations for 2010: https://sedac.ciesin.columbia.edu/data/set/gpw-v4-basic-demographic-characteristics-rev11/data-download (each file ~2GB).
- Total population in 2000, 2005, 2010, 2015, 2020 (unraked): https://sedac.ciesin.columbia.edu/data/set/gpw-v4-population-count-rev11/data-download (each file ~400MB).

Downloaded datasets should be saved in the following directory structure under `gpw_path`:

~/Data/population/GPW 4.11/ Demographic characteristics gpw-v4-basic-demographic-characteristics-rev11_a000_004_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a005_009_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a010_014_2010_30_sec gpw-v4-basic-demographic-characteristics-rev11_a015_019_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a020_024_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a025_029_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a030_034_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a035_039_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a040_044_2010_30_sec gpw-v4-basic-demographic-characteristics-rev11_a045_049_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a050_054_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a055_059_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a060_064_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a065_069_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a070_074_2010_30_sec gpw-v4-basic-demographic-characteristics-rev11_a075_079_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a080_084_2010_30_sec_tif gpw-v4-basic-demographic-characteristics-rev11_a085plus_2010_30_sec_tif Unraked gpw-v4-population-count-rev11_2000_30_sec_tif gpw-v4-population-count-rev11_2005_30_sec_tif gpw-v4-population-count-rev11_2010_30_sec_tif gpw-v4-population-count-rev11_2015_30_sec_tif gpw-v4-population-count-rev11_2020_30_sec_tif

Value

A data frame formatted as Naomi population dataset.

---

naomi_extract_worldpop

Extract WorldPop raster data

Description

Extract WorldPop raster data

Usage

```
naomi_extract_worldpop(
  areas,
  iso3 = areas$area_id[areas$area_level == 0],
  years = c(2010, 2015, 2020)
)
```

Arguments

areas	Naomi area hierarchy dataset with boundaries.
iso3	ISO3 country code.
years	Years to extract WorldPop data

Details

Raster files are downloaded from the WorldPop FTP. Some files are very large. It is recommended to run this on a fast internet connection.

Value

A data frame formatted as Naomi population dataset

---

plot_area_hierarchy_summary

Plot area hierarchy levels

Description

Plot area hierarchy levels

Usage

```
plot_area_hierarchy_summary(areas, nrow = 1)
```

Arguments

areas	area hierarchy sf object
nrow	number of rows, integer.

Value

A ggplot2 object illustrating the area hierarchy

plot_survey_coordinate_check

Summary plot of survey cluster coordinates outside boundaries

Description

Summary plot of survey cluster coordinates outside boundaries

Usage

```
plot_survey_coordinate_check(
  survey_clusters,
  survey_region_boundaries,
  survey_region_areas
)
```

Arguments

survey_clusters	Survey clusteres dataset.
survey_region_boundaries	Survey region boundaries dataset.

Details

The `survey_region_boundaries` dataset is used to define the scope of what is plotted. A subset of regions can be plotted by subsetting that dataset to the desired range.

Value

A list of grobs, one for each survey.

read_pjnz_region_code

Read Spectrum region code from PJNZ file

Description

Read Spectrum region code from PJNZ file

Usage

```
read_pjnz_region_code(pjnz)
```

Arguments

pjnz	file path to source PJNZ

read_sf_zip — *Read shape file from ZIP*

Description

Read shape file from ZIP

Usage

```
read_sf_zip(zfile, pattern = "shp$")
```

Arguments

zfile	Path to zip file
pattern	Pattern to read files for from zip, defaults to files ending with 'shp'

read_sf_zip_list — *Read Multiple Shape Files in ZIP Archive*

Description

Reads all files in ZIP archive `zfile` matching `pattern` with function `read_fn` and returns as a list.

Usage

```
read_sf_zip_list(zfile, pattern = "\\.shp$", read_fn = sf::read_sf)
```

Arguments

zfile	path to a zip directory
pattern	string pattern passed to `list.files`.
read_fn	function used to read matched files.

read_shiny90_country *Read country from .zip.shiny90 file*

Description

Read country from .zip.shiny90 file

Usage

```
read_shiny90_country(shiny90_zip)
```

Arguments

shiny90_zip	path to .shiny90 export file

Value

Shiny90 country / region name.

recode_naomi1_age_group *Recode age group from Naomi 1 to Naomi 2*

Description

Recode age group from Naomi 1 to Naomi 2

Usage

```
recode_naomi1_age_group(x)
```

Arguments

x	Character vector of age groups in Naomi 1 format

Value

Character vector of age groups in Naomi 2 format

Examples

```
recode_naomi1_age_group(c("15-19", "15+", "00+"))
```

`recode_naomi1_art` *Update ART and ANC programme data set to Naomi 2.0 specifications*

Description

Update ART and ANC programme data set to Naomi 2.0 specifications

Usage

```
recode_naomi1_art(art)

recode_naomi1_anc(anc)
```

Arguments

`art`	Data frame of ART data conforming to Naomi 1.0 schema.
`anc`	Data frame of ANC testing data conforming to Naomi 1.0 schema.

Details

- Rename `current_art` column to `art_current`.
- Recode `year` column to `calendar_quarter` in ART dataset.
- Recode `age_group` column from `15-49` format to `Y015_049`.
- Recode `ancrt_*` columns to `anc_*`.

Value

Data frame of ART data conforming to Naomi 2.0 schema.

`surveys_add_dhs_regvar` *Add REGVAR to surveys dataset*

Description

The variable name for the survey region variable is sourced from the DHS survey boundaries datasets sourced by `create_survey_boundaries_dhs()`. Utility function to merge survey region variable name to `surveys` dataset from `survey_region_boundaries` dataset.

Usage

```
surveys_add_dhs_regvar(surveys, survey_region_boundaries)
```

Arguments

`surveys`	surveys dataset, data.frame.
`survey_region_boundaries`	survey_region_boundaries dataset, sf object.

Details

This will throw an error if the REGVAR is not unique to each survey_id within the `survey_region_boundaries` dataset.

Value

The surveys data.frame

`validate_naomi_population` *Validate naomi population dataset*

Description

Validate naomi population dataset

Usage

```
validate_naomi_population(population, areas, area_level)
```

Arguments

`area_level`	area level(s) at which population is supplied

Details

Check that:

- Column names match schema
- Population stratification has exactly area_id / sex / age_group for each year data are supplied

Value

Invisibly TRUE or raises error.

`validate_survey_region_areas` *Validation of mapping to survey region areas*

Description

Validation of mapping to survey region areas

Usage

```
validate_survey_region_areas(
  survey_region_areas,
  survey_region_boundaries,
  warn = FALSE
)
```

Arguments

`survey_region_areas`	Allocation of areas to survey regions, returned by `allocate_areas_survey_regions()`.
`survey_region_boundaries`	survey_region_boundaries dataset created by `create_survey_boundaries_dhs()`.
`warn`	Raise a warning instead of an error (default `FALSE`)

Details

Conducts checks on `survey_region_areas`:

- All areas have been mapped to a survey region in each survey.
- All survey regions contain at least one area. Otherwise no clusters could have come from that survey region.

Passing these checks does not confirm the mapping is accurate, but these checks will flag inconsistencies that need cleaning.

Value

invisibly TRUE or raises an error.

`write_sf_shp_zip` *Save sf object to zipped ESRI .shp file*

Description

Save an sf object as a zipped archive with the four ESRI shape file components `.shp`, `.dbf`, `.prj`, `.shx`. This wraps `sf::write_sf()`.

Usage

```
write_sf_shp_zip(obj, zipfile, overwrite = FALSE)
```

Arguments

`obj`	an object of class `sf`.
`zipfile`	path to write zip output file. Must have file extension .zip.
`overwrite`	logical whether to overwrite `zipfile` if it already exists.

Value

Return value of `file.copy()`, `TRUE` if file successfully written.

Examples

```
nc <- read_sf(system.file("shape/nc.shp", package="sf"))
write_sf_shp_zip(nc, "nc.zip")
```

Index