Package 'SMMT'

May 7, 2026

Title The Swiss Municipal Data Merger Tool Maps Municipalities Over Time

Version 1.2.0

Description In Switzerland, the landscape of municipalities is changing rapidly mainly due to mergers. The Swiss Municipal Data Merger Tool automatically detects these mutations and maps municipalities over time, i.e. municipalities of an old state to municipalities of a new state. This functionality is helpful when working with datasets that are based on different spatial references. The package's idea and use case is discussed in the following article: <doi:10.1111/spsr.12487>.

Imports dplyr, XML, tibble, curl, rvest, xml2

Suggests testthat, roxygen2, knitr, rmarkdown

URL https://github.com/ValValetl/SMMT

BugReports https://github.com/ValValetl/SMMT/issues

VignetteBuilder knitr

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation no

Author Valentin Knechtl [aut, cre] (ORCID: <https://orcid.org/0000-0002-8545-8883>)

Maintainer Valentin Knechtl <valentinknechtl@gmail.com>

Repository CRAN

Date/Publication 2024-05-20 10:40:02 UTC

Contents

`date_of_last_update` *Get date of last municipal inventory update*

Description

Obtain the most recent change in the municipal inventory database. Most but not all municipal mutations are made at first of january.

Usage

```
date_of_last_update(mutations)
```

Arguments

`mutations`	A tibble with municipality mutations (as created by `import_CH_municipality_inventory`)

Value

A `Date` object of length one.

`download_municipality_inventory` *Download municipality inventory*

Description

This functions downloads and extracts the municipality inventory form a defined online source.

Usage

```
download_municipality_inventory(
  url = get_current_url(),
  path = getwd(),
  verbose = TRUE
)
```

Arguments

`url`	Character vector of length one. Link to the zip file containing the municipality inventory.
`path`	Character vector of length one. Destination of extracted xml file.
`verbose`	Get a message after download about the content of the inventory.

Value

Character vector of length one. File path to the extracted XML file.

`filter_date` *Filter by date*

Description

Filter for existing municipalities at a specific point in time.

Usage

```
filter_date(tbl, date)
```

Arguments

`tbl`	A tibble
`date`	A `Date` object of length one.

Value

A tibble which is a subset of `tbl`

`get_current_url` *Get URL of current XML data set*

Description

Extract the url from the static source web page.

Usage

```
get_current_url()
```

Value

Returns URL to municipality inventory

`get_irreversible_municipality_mutations` *Get irreversible municipality mutations*

Description

This function detects irreversible mutations.

Usage

```
get_irreversible_municipality_mutations(mutations)
```

Arguments

`mutations`	A tibble with municipality mutations (as created by `import_CH_municipality_inventory`)

Details

Irreversible mutations are defined as mutations during which territories are split up. There are different types of irreversible mutations drawn from the below cited document. In contrast, normal mutations signify a simple merging of territory which accounts for most of the mutations in Switzerland since 1960 whereas irreversible mutations occurred only rarely. The aim of this function is to filter for these irreversible mutations. These can then be treated separatly.

Definitions for different types of territory split ups are based on: Erläuterungen und Anwendungen - Historisierte Gemeindeverzeichnis der Schweiz (2017).

Value

A tibble with all the instances of irreversibe mutations. The irreversibility cause is part of the output.

Examples

```
 mutations <- structure(list(hist_id = c(11320L, 13668L, 13669L),
district_hist_id = c(10024L, 10024L, 10024L),
kanton_abbr = c("AG", "AG", "AG"),
bfs_nr = c(4061L, 4061L, 4084L),
name = c("Arni-Islisberg", "Arni (AG)", "Islisberg"),
admission_nr = c(1000L, 1481L, 1481L),
admission_mode = c(20L, 21L, 21L),
admission_date = structure(c(-3653, 4748,  4748),
class = c("Date")),
abolition_nr = c(1481L, NA, NA),
abolition_mode = c(29L, NA, NA),
abolition_date = structure(c(4747, NA, NA),
class = c("Date")),
change_date = structure(c(4747, 4748, 4748),
class = c("Date"))),
row.names = c(NA, -3L), class = c("tbl_df", "tbl", "data.frame"))
```

```
irreversible_mutations <- get_irreversible_municipality_mutations(mutations)
```

`import_CH_municipality_inventory` *Import the Swiss Municipality inventory*

Description

This function imports the Swiss municipality inventory from the raw XML resource into R as a `tibble`. The imported table is the basis to map the Swiss municipalities from an old to a new state (see `map_old_to_new_state`).

Usage

```
import_CH_municipality_inventory(file_path)
```

Arguments

`file_path`	Character vector of length one. It contains the file path to the Swiss municipality inventory XML file.

Details

This imported Swiss municipality inventory is a database with the complete mutation history that occured since 01.01.1960. The Swiss municipality inventory is made available by the Federal Statistical Office and updated regularly to keep track of new mutations.

Download

See BfS webpage for infos about Swiss municipality inventory: Historisiertes Gemeindeverzeichnis

Value

A list with two tables in the form of tibble objects.

1. Municipality mutations.
2. Canton mutations

See Also

`map_old_to_new_state`

`map_old_to_new_state` *Map municipalities of old state to municipalities of new state*

Description

This function maps the Swiss municipalities of an old state to municipalities of a new state.

Usage

```
map_old_to_new_state(mutations, state_old, state_new)
```

Arguments

`mutations`	A tibble containing the municipality mutations inventory (see `import_CH_municipality_inventory`)
`state_old`	A Date object vector of length one containing the date of the old state.
`state_new`	A Date object vector of length one containing the date of the new state.

Details

Approach

1. Download the Swiss municipality inventory
2. Import it into R workspace with `import_CH_municipality_inventory`
3. Set the old state and the new state (see example)
4. Get the mapping table with this function

Example Daettwil / Baden

On 1.1.1962 Daettwil (Bfs Nr. 4025) merged with Baden (Bfs Nr. 4021). Let's define

- old_state <- as.Date("1961-01-01")
- new_state <- as.Date("1963-01-01")
- Result:

bfs_nr_new	name_new	bfs_nr_old	name_old
4021	Baden	4021	Baden
4021	Baden	4025	Daettwil

Value

A list with 4 elements:

1. mapped: A tibble with the mapped municipalities
2. unmapped: A tibble with the unmapped municipalities
3. state_old: see above
4. state_new: see above

Examples

```
mutations <- structure(list(hist_id = c(11227L, 11240L, 13189L),
district_hist_id = c(10025L, 10025L, 10025L),
kanton_abbr = c("AG", "AG", "AG"),
bfs_nr = c(4025L, 4021L, 4021L),
name = c("Daettwil", "Baden", "Baden"),
admission_nr = c(1000L, 1000L, 1004L),
admission_mode = c(20L, 20L, 26L),
admission_date = structure(c(-3653, -3653, -2922),
class = c("Date")),
abolition_nr = c(1004L, 1004L, NA),
abolition_mode = c(29L, 26L, NA),
abolition_date = structure(c(-2923, -2923, NA),
class = c("Date")),
change_date = structure(c(-2923, -2923, -2922), class = c("Date"))),
row.names = c(NA, -3L), class = c("tbl_df", "tbl", "data.frame"))

mapping_object <- map_old_to_new_state(mutations,
as.Date("1961-01-01"), as.Date("1963-01-01"))
```

---

most_recent_changes *Most recent changes*

Description

Returns a table with the most recently changed municipalities.

Usage

```
most_recent_changes(mutations)
```

Arguments

mutations	A tibble with municipality mutations (as created by import_CH_municipality_inventory)

---

municipality_counter *Municipality counter*

Description

Count the municipalities for a set of dates. Either at the national or cantonal level. See vignette for details.

Usage

```
municipality_counter(
  mutations,
  dates,
  geo_level = "ch",
  include_cant_lakes = FALSE
)
```

Arguments

mutations	A tibble containing the municipality mutations inventory (see import_CH_municipality_inventory)
dates	A Date object vector
geo_level	Either "ch" or "cantons".
include_cant_lakes	Boolean, TRUE to also include lakes in the count.

Value

A tibble with the municipality count per date and specified geography.

Note

All entities that have a bfs nr are counted (e.g. also Gemeindefreie Gebiete). This is not exactly what the BfS does in the webtool Applikation der Schweizer Gemeinden. However, it is not possible to distinguish "Gemeinden" und "Gemeindefreie Gebiete" generically, based on the information in the Gemeindeverzeichnis.

---

mutation_count *Mutation count*

Description

Count number of mutations in a given time period

Usage

```
mutation_count(
  mutations,
  start_date,
  end_date = Sys.Date(),
  territorial_changes_only = FALSE
)
```

Arguments

mutations	A tibble containing the municipality mutations inventory (see import_CH_municipality_inventory)
start_date	Date vector (incl)
end_date	Date vector (excluded)
territorial_changes_only	boolean. FALSE if all mutations should be considered. TRUE if mutations that have an effect on the municipal territory only should be considered. FALSE includes name changes, Bezirk number changes etc.

Details

Approach

1. Download the Swiss municipality inventory
2. Import it into R workspace with import_CH_municipality_inventory
3. Set the old state and the new state (see example)
4. Get the mapping table with this function

Example Daettwil / Baden

On 1.1.1962 Daettwil (Bfs Nr. 4025) merged with Baden (Bfs Nr. 4021). Let's define

- old_state <- as.Date("1961-01-01")
- new_state <- as.Date("1963-01-01")
- Result:

bfs_nr_new	name_new	bfs_nr_old	name_old
4021	Baden	4021	Baden
4021	Baden	4025	Daettwil

Value

A list with 4 elements:

1. mapped: A tibble with the mapped municipalities
2. unmapped: A tibble with the unmapped municipalities
3. state_old: see above
4. state_new: see above

Examples

```
mutations <- structure(list(hist_id = c(11227L, 11240L, 13189L),
district_hist_id = c(10025L, 10025L, 10025L),
kanton_abbr = c("AG", "AG", "AG"),
bfs_nr = c(4025L, 4021L, 4021L),
name = c("Daettwil", "Baden", "Baden"),
admission_nr = c(1000L, 1000L, 1004L),
admission_mode = c(20L, 20L, 26L),
```

```
admission_date = structure(c(-3653, -3653, -2922),
class = c("Date")),
abolition_nr = c(1004L, 1004L, NA),
abolition_mode = c(29L, 26L, NA),
abolition_date = structure(c(-2923, -2923, NA),
class = c("Date")),
change_date = structure(c(-2923, -2923, -2922), class = c("Date"))),
row.names = c(NA, -3L), class = c("tbl_df", "tbl", "data.frame"))

mapping_object <- map_old_to_new_state(mutations,
as.Date("1961-01-01"), as.Date("1963-01-01"))
```

`territorial_mutation_count` *Territorial mutation coutn*

Description

Count number of mutations in a given time period

Usage

```
territorial_mutation_count(mutations, start_date, end_date = Sys.Date())
```

Arguments

`mutations`	A tibble containing the municipality mutations inventory (see `import_CH_municipality_inventory`)
`start_date`	Date vector (incl)
`end_date`	Date vector (excluded)

Details

Approach

1. Download the Swiss municipality inventory
2. Import it into R workspace with `import_CH_municipality_inventory`
3. Set the old state and the new state (see example)
4. Get the mapping table with this function

Example Daettwil / Baden

On 1.1.1962 Daettwil (Bfs Nr. 4025) merged with Baden (Bfs Nr. 4021). Let's define

- old_state <- as.Date("1961-01-01")
- new_state <- as.Date("1963-01-01")

- Result:

bfs_nr_new	name_new	bfs_nr_old	name_old
4021	Baden	4021	Baden
4021	Baden	4025	Daettwil

Value

A list with 4 elements:

1. mapped: A tibble with the mapped municipalities
2. unmapped: A tibble with the unmapped municipalities
3. state_old: see above
4. state_new: see above

Examples

```
mutations <- structure(list(hist_id = c(11227L, 11240L, 13189L),
district_hist_id = c(10025L, 10025L, 10025L),
kanton_abbr = c("AG", "AG", "AG"),
bfs_nr = c(4025L, 4021L, 4021L),
name = c("Daettwil", "Baden", "Baden"),
admission_nr = c(1000L, 1000L, 1004L),
admission_mode = c(20L, 20L, 26L),
admission_date = structure(c(-3653, -3653, -2922),
class = c("Date")),
abolition_nr = c(1004L, 1004L, NA),
abolition_mode = c(29L, 26L, NA),
abolition_date = structure(c(-2923, -2923, NA),
class = c("Date")),
change_date = structure(c(-2923, -2923, -2922), class = c("Date"))),
row.names = c(NA, -3L), class = c("tbl_df", "tbl", "data.frame"))

mapping_object <- map_old_to_new_state(mutations,
as.Date("1961-01-01"), as.Date("1963-01-01"))
```

Index