Package: il.cbs.muni (via r-universe)

May 14, 2026

Title Utility Functions to Work with Israeli Central Bureau of Statistics Municipal Data

Version 0.1.0

Description Analyst oriented utility functions to handle the different quirks of the Israeli CBS municipal data, harmonize id's and bring together data points from different years.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr, janitor, purrr, readxl, rlang, stringr, tidyr

Suggests httr2, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/matanhakim/il.cbs.muni, https://matanhakim.github.io/il.cbs.muni/

BugReports https://github.com/matanhakim/il.cbs.muni/issues

Depends R (>= 4.1.0)

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-02-10 14:45:36 UTC

RemoteUrl https://github.com/matanhakim/il.cbs.muni

RemoteRef HEAD

RemoteSha 43a6ec44d6c0b8ebf01c2ce6dfd0eb36d367dcc0

Contents

`clean_name` *Clean a string from characters unwanted in (mostly) yishuv names*

Description

Clean a string from characters unwanted in (mostly) yishuv names

Usage

```
clean_name(name)
```

Arguments

`name`	a character vector

Value

a character vector without leading and trailing white space, and with characters that are only letters and digits, excluding the following: `'-()\"`

Examples

```
clean_name("test-123_test*456")
x <- read_cbs_yishuv(system.file("extdata", "bycode2021.xlsx",
 package = "il.cbs.muni"))[[1]][c(153, 342)]
x
clean_name(x)
```

combine_cbs_muni *Combine municipalities data frames from different sheets*

Description

This function is a wrapper around `read_cbs_muni()` to help in combining data for cities, local councils and regional councils. From 2015 and earlier, the Israeli CBS publishes municipal data on different sheets and formats for cities and local councils, and for regional councils. This function enables the user to combine the two data frames for selected columns. It is up to the user to take care of the specific match between specific columns.

Usage

```
combine_cbs_muni(
  path,
  year,
  cols_city,
  cols_rc,
  data_domain = c("physical", "budget"),
  col_names = NULL,
  col_names_from = c("city_lc", "rc")
)
```

Arguments

`path`	A character vector of length 1, denoting the local file path to the municipal data file. A full list of available files by the CBS is at the relevant CBS page.
`year`	A numeric vector of length 1 denoting which year the data file pointed in `path` is for. Currently supporting 2003-2015, since before 2003 the data structure is very different, and after 2015 the file is already combined.
`cols_city`	<tidy-select> Columns to keep from the cities and local councils sheet.
`cols_rc`	<tidy-select> Columns to keep from the regional councils sheet. Should be in the same order of desired columns as in `cols_city`, since the columns are matched by order.
`data_domain`	A character vector of length 1, one of `c("physical", "budget")`. Every Excel municipal data file has a few different data domains, most notably physical and population data, and budget data.
`col_names`	A character vector containing the new column names of the output tibble. If `NULL` then the tibble uses the original column names. Must be the same length as the number of columns picked in `cols`. If not `NULL`, overrides the choice in `col_names_from`.
`col_names_from`	A character vector of length 1, one of `c("city_lc", "rc")`. Denotes which column names should be kept - those from the cities and local councils sheet, or those from the regional councils sheet.

Value

A tibble with municipal data for a specific year, with the columns from `cols_city` and `cols_rc` bound by rows and matched by order of columns. Every row is a municipality and every column is a different variable for this municipality in that year. Be advised all columns are of type character, so you need to parse the data types yourself at will. Column names are merged from the relevant headers, and only single whitespaces are kept. Rows with more than 90% empty cells (usually rows with non-data notes) are removed.

Examples

```
df_1 <- combine_cbs_muni(
  system.file("extdata", "2009.xls", package = "il.cbs.muni"),
  year = 2009,
  cols_city = c(1:7, 11),
  cols_rc = c(1:7, 25)
)

df_1 |>
  dplyr::glimpse()

df_2 <- combine_cbs_muni(
  system.file("extdata", "2009.xls", package = "il.cbs.muni"),
  year = 2009,
  cols_city = c(1:12),
  cols_rc = c(1:12),
  data_domain = "budget",
  col_names_from = "rc"
)

df_2 |>
  dplyr::glimpse()
```

`modify_muni_id` *Modify a municipal id depending on the municipal status and yishuv id*

Description

In the context of working with cbs municipal and yishuvim data, there is a difference in the treatment of a yishuv and a municipality. In the cases of cities and local councils the yishuv id and the municipality id are the same. But in the case of regional councils, the municipality id is for the regional council, while every yishuv within that municipality has a different yishuv id. The Israeli CBS uses a concept of "municipal status" to differentiate between the two. A municipal status of "0" represents a city, and "99" represents a local council. Every other 2-digit number is the municipal id of a regional council. This function modifies a municipal status based on itself and the yishuv_id. It returns the correct municipal id after accounting for the "0" or "99" values.

Usage

```
modify_muni_id(muni_id, yishuv_id)
```

Arguments

muni_id	A character or numeric vector indicating the municipal status, where "0" or "99" represents a city or a local council, and every other two-digit number or character represents a regional council.
yishuv_id	A character or numeric vector representing the yishuv id. Should be 4 digits long according to the il.verse conventions.

Value

A character vector with 4 digits municipal id for cities and local councils and 2 digits municipal id for regional councils.

Examples

```
muni_id <- c(0, 99, 1, 2)
yishuv_id <- c("0001", "1000", "1234", "1567")
modify_muni_id(muni_id, yishuv_id)
```

---

pad_yishuv_id *Pad a yishuv id to a 4-digit character vector*

Description

Pad a yishuv id to a 4-digit character vector

Usage

```
pad_yishuv_id(yishuv_id)
```

Arguments

yishuv_id	a character vector containing only characters and numbers, where each element is no longer than 4 characters or digits.

Value

A character or numeric vector, where each element is 4 characters long, containing only numbers and left-padded with 0's.

Examples

```
x <- c(1, "23", "4000", 5600)
pad_yishuv_id(x)
```

`read_cbs_index` *Read a CBS index data file to a tibble*

Description

This function is a wrapper around `readxl::read_excel()`, reading a specific CBS index data file for a specific year and a specific data domain. Its added value is in its pre-defined parameters for every year and its specific quirks in the Excel headers. For advanced users,the full set of options is available with `il.cbs.muni:::df_cbs_index_params`.

Usage

```
read_cbs_index(
  path,
  year,
  index_type = c("ses", "peri"),
  unit_type = c("muni", "yishuv", "sa"),
  cols = NULL,
  col_names = NULL
)
```

Arguments

`path`	A character vector of length 1, denoting the local file path to the CBS index data file. A full list of available files by the CBS is at the relevant CBS page for either Socio-Economic Status (SES) or for peripheral level.
`year`	A numeric vector of length 1 denoting which year the data file pointed in `path` is for. Be aware that the year in question is the year **the data is for**, not the year **the data was published in**.
`index_type`	A character vector of length 1, one of `c("ses", "peri")`.
`unit_type`	A character vector of length 1, one of `c("muni", "yishuv", "sa")`. • `"muni"` - every row is a municipality. • `"yishuv"` - every row is a yishuv. In some years and indices this includes all yishuvim, in others only yishuvim within regional councils. • `"sa"` - every row is a statistical area within a city or local council.
`cols`	<tidy-select> Columns to keep. The default `NULL` keeps all columns.
`col_names`	A character vector containing the new column names of the output tibble. If `NULL` then the tibble uses the original column names. Must be the same length as the number of columns picked in `cols`.

Value

A tibble with CBS index data for a specific year, where every row is a `unit_type` and every column is a different variable for this `unit_type` in that year. Be advised all columns are of type character, so you need to parse the data types yourself at will. Column names are merged from the relevant headers, and only single whitespaces are kept. Rows with more than 90% empty cells (usually rows with non-data notes) are removed.

Examples

```
read_cbs_index(
  path = system.file("extdata", "24_22_375t1.xlsx", package = "il.cbs.muni"),
  year = 2019,
  index_type = "ses",
  unit_type = "muni"
) |>
  dplyr::glimpse()
```

`read_cbs_muni` *Read a municipal data file to a tibble*

Description

This function is a wrapper around `readxl::read_excel()`, reading a specific municipal data file for a specific year and a specific data domain. Its added value is in its use of `row_to_names_fill()` and its pre-defined parameters for every year and its specific quirks in the Excel headers. For advanced users, the full set of options is available with `il.cbs.muni:::df_cbs_muni_params`.

Usage

```
read_cbs_muni(
  path,
  year,
  muni_type = c("all", "city_lc", "rc"),
 data_domain = c("physical", "budget", "summary", "labor_force_survey", "social_survey"),
  cols = NULL,
  col_names = NULL
)
```

Arguments

`path`	A character vector of length 1, denoting the local file path to the municipal data file. A full list of available files by the CBS is at the relevant CBS page.
`year`	A numeric vector of length 1 denoting which year the data file pointed in `path` is for. Currently supporting only 2003 and later, since before 2003 the data structure is very different.
`muni_type`	A character vector of length 1, one of `c("all", "city_lc", "rc")`. Since 2016, all municipal types are bundled together in the same sheets, but before 2016 there are different sheets for cities and local councils (`"city_lc"`) and regional councils (`"rc"`). This parameter chooses which sheet you would read.
`data_domain`	A character vector of length 1, one of `c("physical", "budget", "summary", "labor_force_survey", "social_survey")`. Every Excel municipal data file has a few different data domains, most notably physical and population data, and budget data.
`cols`	<tidy-select> Columns to keep. The default `NULL` keeps all columns.

col_names — A character vector containing the new column names of the output tibble. If NULL then the tibble uses the original column names. Must be the same length as the number of columns picked in cols.

Value

A tibble with municipal data for a specific year, where every row is a municipality and every column is a different variable for this municipality in that year. Be advised all columns are of type character, so you need to parse the data types yourself at will. Column names are merged from the relevant headers, and only single whitespaces are kept. Rows with more than 90% empty cells (usually rows with non-data notes) are removed.

Examples

```
df <- read_cbs_muni(
  system.file("extdata", "p_libud_2021.xlsx", package = "il.cbs.muni"),
  year = 2021,
  data_domain = "physical"
)

df |>
  dplyr::select(1:15) |>
  dplyr::glimpse()
```

---

read_cbs_yishuv *Read a yishuvim data file to a tibble*

Description

This function is a wrapper around readxl::read_excel(), reading a specific yishuvim data file or a part of it. A yishuv, or a point of residence, is a geographically defined place where people live. Some yishuvim are municipalities, in the case of of cities and local councils, but most are not. most yishuvim are part of municipalities that are regional councils. Also, some yishuvim are not themselves and are not part of a municipality, like some Bedouin places in southern Israel, some industry areas, Mikveh Israel, and more.

Usage

```
read_cbs_yishuv(path, cols = NULL, col_names = NULL)
```

Arguments

path — A character vector of length 1, denoting the local file path to the yishuvim data file. A full list of available files by the CBS is at the relevant CBS page.

cols — <tidy-select> Columns to keep. The default NULL keeps all columns.

col_names — A character vector containing the new column names of the output tibble. If NULL then the tibble uses the original column names. Must be the same length as the number of columns picked in cols.

Value

A tibble with yishuvim data for a specific year, where every row is a yishuv and every column is a different variable for this yishuv in that year. Be advised all columns are of type character, so you need to parse the data types yourself at will. Column names are cleaned so only single whitespaces are kept.

Examples

```
library(dplyr)
read_cbs_yishuv(system.file("extdata", "bycode2021.xlsx", package = "il.cbs.muni")) |>
  dplyr::glimpse()

read_cbs_yishuv(
  system.file("extdata", "bycode2021.xlsx", package = "il.cbs.muni"),
  cols = c(1, 2, 5, 13)
) |>
  mutate(across(2, pad_yishuv_id)) |>
  glimpse()
```

`read_muni_id` *Read municipalities id's and names*

Description

Israeli municipalities have different id's and sometimes even different names across different government organizations. This function allows you to read different municipality id's and names , so interchanging between the different specifications would be easier.

Usage

```
read_muni_id(id_types = c("muni", "edu", "tax"), include_names = FALSE)
```

Arguments

`id_types`	A character vector of length between 1 and 3, containing at least one (or two, or all of) of the possible values. id's (and possibly names) of municipalities are kept for the selected sources: • `"muni"` is for CBS id's and (cleaned) names. • `"edu"` is for Ministry of Education municipal symbol ("Semel Reshut" in Hebrew) • `"tax"` is for Israel Tax Authority municipal id (also known as a "H.P. number")
`include_names`	A logical vector of length 1, denoting if the names of municipalities (for each of the `id_types` chosen) should be included. Be aware that some municipal names might differ between different agencies.

Value

A tibble, where every row is a municipality and the columns include id's (and possibly names) of the municipalities from the chosen agencies.

Examples

```
read_muni_id() |>
  dplyr::glimpse()

read_muni_id(id_types = c("muni", "edu"), include_names = TRUE) |>
  dplyr::glimpse()
```

---

row_to_names_fill *Elevate rows to be the column names of a data.frame and fill row-wise if needed*

Description

Casts data from rows to the column names of a data frame, with the option to fill missing values row-wise. This utility is helpful in the case of merged cells in Microsoft Excel, where the merged range has data only in the first (unmerged) cell. This function is similar to `janitor::row_to_names()`, with the exception of the fill utility.

Usage

```
row_to_names_fill(
  data,
  row_number,
  fill_missing = TRUE,
  remove_row = TRUE,
  remove_rows_above = TRUE,
  sep = "_"
)
```

Arguments

`data`	A data frame.
`row_number`	A numeric vector with he row indices of `data` containing the variable names. Allows for multiple rows input as a numeric vector. If multiple rows, values in the same column would be pasted with the `sep` argument as a separator. NAs are ignored.
`fill_missing`	A logical vector of length 1 or of length `length(row_number)`. Every value in the vector denotes for the matching row in `data[row_number, ]` if the row should fill missing values (from left to right). If `TRUE` for a row, all missing values following a non-missing value will be replaced with that preceding non-missing value.

remove_row
: A logical vector of length 1, denoting if the row `row_number` should be removed from the resulting data.frame.

remove_rows_above
: A logical vector of length 1, denoting if the rows above `row_number` - that is, between `1:(row_number-1)` - should be removed from the resulting data.frame, in the case that `row_number != 1`.

sep
: A character vector of length 1 to separate the values in the case of multiple rows input to `row_number`.

Value

A data frame (class `data.frame`) with the same structure as the input `data`, but with column names derived from the specified row(s). The returned data frame has the same number of columns as the input, with rows removed according to the `remove_row` and `remove_rows_above` parameters. All data types and values are preserved from the original data frame.

Examples

```
df <- data.frame(
  a = 1:6,
  b = rep(c("x", NA), 3),
  c = letters[1:6]
)

df

row_to_names_fill(df, 2:3)
row_to_names_fill(df, 2:3, sep = ".")
row_to_names_fill(df, 2:4, fill_missing = c(TRUE, FALSE, FALSE))
```

Index