Package ‘stlcsb’

February 22, 2019

Type Package

Title Tidy Manipulation of CSB Data for St. Louis

Version 0.1.2

Description The Citizens' Service Bureau of the City of St. Louis is a clearing house for non-emergency service requests. This package provides functions for downloading, categorizing problem requests, cleaning and subsetting CSB data, and projecting the data using the x and y coordinates included with CSB data releases.

Depends R (>= 3.3)

License GPL-3

URL https://github.com/slu-openGIS/stlcsb

BugReports https://github.com/slu-openGIS/stlcsb/issues

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports dplyr, lubridate, purrr, readr, readxl, rlang, rvest, sf, stringr, tibble, tools, xml2

Suggests ggplot2, knitr, mapview, rmarkdown, testthat, covr

VignetteBuilder knitr

NeedsCompilation no

Author Christopher Prener [aut, cre] (<https://orcid.org/0000-0002-4310-9888>), Branson Fox [aut] (<https://orcid.org/0000-0002-4361-2811>)

Maintainer Christopher Prener <chris.prener@slu.edu>

Repository CRAN

Date/Publication 2019-02-22 00:20:02 UTC

R topics documented:

cat *Lookup Tables Used for Categorization of Problem Codes*

Description

Any data object from the `stlcsb` package that starts with `cat_` is a lookup table. These tables were hand-made - contact the package maintainers with questions or concerns. These tables are used internally in any function that categorizes by problem code. They are available to the user as a matter of convenience and transparency.

Usage

```
cat_admin

cat_animal

cat_construction

cat_debris

cat_degrade

cat_disturbance

cat_event

cat_health

cat_landscape

cat_law

cat_maintenance
```

`cat_nature`

`cat_road`

`cat_sewer`

`cat_traffic`

`cat_vacant`

`cat_waste`

Format

An object of class `character` of length 78.

Value

A named character vector listing the specific problem codes associated with each category.

`csb_canceled` *Remove Canceled Calls for Service*

Description

Subsets data to remove any call with a date and time canceled.

Usage

```
csb_canceled(.data, var, drop = TRUE)
```

Arguments

`.data`	A tibble or data frame
`var`	Name of the column containing cancellation timestamps
`drop`	A logical scalar; if `TRUE`, removes the now empty column that had contained cancellation date and time, otherwise if `FALSE` the empty column is retained.

Value

Returns a tibble with the rows containing dates and times for the given variable removed.

Examples

```
csb_canceled(january_2018, var = "datecancelled")
csb_canceled(january_2018, var = "datecancelled", drop = FALSE)
```

csb_categorize *Categorize CSB Call Types*

Description

csb_categorize provides general categories for the CSB data based on problem code. These were created based on a review of the call data in mid-2018.

Usage

```
csb_categorize(.data, var, newVar)
```

Arguments

.data	A tibble or data frame
var	Name of existing column containing problem codes
newVar	Name of output variable to be created with category string

Value

Returns a tibble with the string vector added as a new variable.

Examples

```
csb_categorize(january_2018, var = problemcode, newVar = Category)
```

csb_date_filter *Filter Calls Based on Date of Call*

Description

csb_date_filter filters dates to return only the specified date elements. For example, data can be returned for specific months, years, or portions of months

The month argument can be one of several types. Types cannot be mixed. A numeric argument specifying month is acceptable. Character entry can be one of either 3 letter abbreviations or full month name. Capitalization does not matter.

Usage

```
csb_date_filter(.data, var, day, month, year)
```

Arguments

`.data`	A tibble or data frame
`var`	A name of column containing date data
`day`	A numeric vector of day(s) to include.
`month`	A numeric/character vector of month(s) to include. See description for more information on alternate entry formats.
`year`	A numeric vector of years(s) to include (2 or 4 digit)

Value

Returns a filtered version of the input data based on specified date arguments.

Examples

```
csb_date_filter(january_2018, datetimeinit, day = 1)
csb_date_filter(january_2018, datetimeinit, day = 1:15, month = 1)
csb_date_filter(january_2018, datetimeinit, month = "January", year = 09)
csb_date_filter(january_2018, datetimeinit, month = c("jan", "feb", "Mar", "Apr"), year = 2009)
csb_date_filter(january_2018, datetimeinit, day = 1:15, month = 1:6, year = 08:13)
```

`csb_date_parse` *Parse CSB Date and Time Variables*

Description

`csb_date_parse` is used to parse out dates into day, month, and year elements.

Usage

```
csb_date_parse(.data, var, day, month, year, drop = FALSE)
```

Arguments

`.data`	A tibble or data frame
`var`	name of column containing date data
`day`	Optional; returns a named column with parsed day
`month`	Optional; returns a named column with parsed month
`year`	Optional; returns a named column with parsed year
`drop`	A logical scalar; if `TRUE`, removes the original column that had contained date and time data, otherwise if `FALSE` the original column is retained.

Value

Returns a tibble with new columns containing parsed date information

Examples

```
csb_date_parse(january_2018, datetimeinit, dayInit)
csb_date_parse(january_2018, datetimeinit, dayInit, monthInit)
csb_date_parse(january_2018, datetimeinit, month = monthInit)
csb_date_parse(january_2018, datetimeinit, month = monthInit, year = yearInit)
csb_date_parse(january_2018, datetimeinit, dayInit, monthInit, yearInit, drop = TRUE)
```

csb_filter *Subset Based on Call Categories*

Description

`csb_filter` returns observations that match any combination of the predefined categories that are created

Usage

```
csb_filter(.data, var, category)
```

Arguments

`.data` A tibble or data frame

`var` name of the column containing original problem code data

`category` a vector with the unquoted name(s) of the category(s) for the function to return. You can also explicitly state quoted problemcode(s). Valid categories are: admin, animal, construction, debris, degrade, disturbance, event, health, landscape, law, maintenance, nature, road, sewer, traffic, vacant, and waste. If categories are listed, each should be preceded by 'cat_'.

Value

`csb_filter` returns data with an additional variable for an intelligible category for CSB requests.

Examples

```
csb_filter(january_2018, var = problemcode, category = cat_vacant)
csb_filter(january_2018, var = problemcode, category = c(cat_waste, cat_debris))
csb_filter(january_2018, var = problemcode, category = "WTR-COMING-UP")
```

csb_get_data *Download CSB Data from the City of St. Louis*

Description

csb_get_data provides direct access to a compiled version of the CSB's data release via the City of St. Louis website. These data are provided with no warranty from either the City of St. Louis or the package developers.

Usage

```
csb_get_data(tidy = TRUE, years, ...)
```

Arguments

tidy	A logical scalar; if TRUE, variable names will be converted to lower case and reordered. Two variables with incomplete data - problem city (PROBCITY) and problem zip code (PROBZIP) - are dropped to limit use of memory. This mirrors the functionality of csb_load_variables.
years	Optional; if included, data not in the specified years will be excluded from the returned object.
...	Additional testing options; not for production use

Value

Returns a tibble with all CSB calls for service.

Examples

```
## Not run:
csb <- csb_get_data()
csb <- csb_get_data(tidy = FALSE)
csb <- csb_get_data(years = 2009:2018)
csb <- csb_get_data(years = 2018)

## End(Not run)
```

`csb_last_update` *Date of Last CSB Data Update from the City of St. Louis*

Description

Data are updated by the City of St. Louis on their open data site on a weekly basis. This function returns the date of the last update.

Usage

```
csb_last_update()
```

Value

A string scalar containing the date of last update.

Examples

```
## Not run:
last_update <- csb_last_update()

## End(Not run)
```

`csb_load_variables` *Load CSB Variable Definitions*

Description

Provides direct access to the CSB's variable definitions, which are available for download from the City of St. Louis's open data site.

Usage

```
csb_load_variables(tidy = TRUE)
```

Arguments

`tidy`	A logical scalar; if `TRUE`, variable names will be converted to lower case and reordered. Two variables with incomplete data - problem city (`PROBCITY`) and problem zip code (`PROBZIP`) - are dropped to limit use of memory. This mirrors the functionality of csb_get_data.

Value

A tibble containing variable names and definitions.

See Also

City of St. Louis Open Data, City of St. Louis CSB Data

Examples

```
csb_load_variables()
csb_load_variables(tidy = FALSE)
```

`csb_missingXY` *Identifying Calls Missing Coordinate Data*

Description

`csb_missingXY` returns a logical vector indicating if an observation is missing in either of the coordinate columns that come with the CSB data.

Usage

```
csb_missingXY(.data, varX, varY, newVar)
```

Arguments

`.data`	A tibble or data frame
`varX`	Name of column containing x coordinate data
`varY`	Name of column containing y coordinate data
`newVar`	Name of new column that is `TRUE` if coordinate data are missing and `FALSE` otherwise.

Value

A tbl with a logical vector appended to it.

Examples

```
csb_missingXY(january_2018, srx, sry, newVar = "missingXY")
```

csb_projectXY *Project Calls for Service Data Using Coordinates*

Description

csb_projectXY converts srx and sry data into a simple features object. You can write a shapefile directly from the output of this function using sf::st_write.

Usage

```
csb_projectXY(.data, varX, varY, crs)
```

Arguments

.data	A tibble or data frame
varX	Name of column containing x coordinate data
varY	Name of column containing y coordinate data
crs	Optional; coordinate reference system for the data to be projected into

Value

Returns a sf object of the input data projected as point data.

Examples

```
# remove missing coordinates prior to projecting
csb <- csb_missingXY(january_2018, srx, sry, newVar = missing)
csb <- dplyr::filter(csb, missing == FALSE)

# project data
csb_projectXY(csb, srx, sry)

# project with a custom crs
csb_projectXY(csb, srx, sry, crs = 4269)
```

csb_vacant *Identify Calls for Service Related to Vacancy*

Description

csb_vacant appends a logical vector indicating TRUE for vacancy related problem codes.

Usage

```
csb_vacant(.data, var, newVar)
```

Arguments

`.data`	A tibble or data frame
`var`	Name of existing column containing problem codes
`newVar`	Name of output variable to be created with vacant logical

Value

Returns a tibble with the logical vector added as a new variable.

Examples

```
csb_vacant(january_2018, var = problemcode, newVar = vacant)
```

`january_2018` *CSB Calls in St. Louis, January 2018*

Description

An example data set containing CSB calls in St. Louis, Missouri during the first week of January, 2018.

Usage

```
data(january_2018)
```

Format

A tibble with 1554 rows and 17 variables:

requestid system generated unique request identifier

datetimeinit date and time the request was initiated

probaddress address of the request

probaddtype A = Parcel, B = Intersection

callertype method used by citizen to report issue (Phone, Web, Twitter, etc)

neighborhood City of St. Louis Neighborhood number (1-79)

ward City of St. Louis Ward number (1-28)

problemcode type of report

description same as problemcode OR slightly more specific

submitto city division responsible for completing the request

status status of the request

dateinvtdone date of investigation-date that work was done, may differ from closing date because of crews using paper copies of requests

datetimeclosed date and time the request was closed

projcompletedate date by which city division should have initial inspection complete, auto-populated based on service level agreements

datecancelled indicates a duplicate, cancelled or entered in error request

srx map coordinate, X-coordinate

sry map coordinate, Y-coordinate

Source

St. Louis Citizens' Service Bureau

Examples

```
str(january_2018)
head(january_2018)
```

Index