Package 'compstatr'

May 14, 2020

Type Package

Title Tools for St. Louis Crime Data

Version 0.2.1

Description Provides a set of tools for creating yearly data sets of St. Louis Metropolitan Police Department (SLMPD) crime data, which are available from January 2008 onward as monthly CSV releases on their website (<http:www.slmpd.org/Crimereports.shtml>). Once data are validated and created (monthly data releases have varying numbers of columns as well as different column names and formats), 'compstatr' also provides functions for categorizing and mapping crimes in St. Louis. The categorization tools that are provided will also work with any police department that uses 5 and 6 digit numeric codes to identify specific crimes. These data provide researchers and policy makers detailed data for St. Louis, which in the last several years has had some of the highest or the highest violent crime rates in the United States.

Depends R (>= 3.4)

License GPL-3

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

BugReports https://github.com/slu-openGIS/compstatr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports dplyr, fs, httr, janitor, lubridate, purrr, rlang, readr, rvest, sf, stringr, tibble, tidyr, xml2

Suggests testthat, knitr, rmarkdown, covr

VignetteBuilder knitr

NeedsCompilation no

Author Christopher Prener [aut, cre], Cree Foeller [aut], Taylor Braswell [com]

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

Repository CRAN

Date/Publication 2020-05-14 17:30:08 UTC

R topics documented:

`cs_address` *Create a Single Address Field*

Description

The street address data in SLMPD releases (either `ILEADSAddress` and `ILEADSStreet` or `CADAddress` and `CADStreet`) are stored in separate columns. In order to faciliate geocoding, this function combines the fields and removes inappropriate characters in the address fields.

Usage

```
cs_address(.data, address, street, newVar)
```

Arguments

`.data`	A tibble or data frame
`address`	Name of address number variable (typically either `ILEADSAddress` or `CADAddress`)
`street`	Name of street name variable (typically either `ILEADSStreet` or `CADStreet`)
`newVar`	Name of new variable to store concatenated address

Value

A copy of the object with a character vector that contains the concatenated street address data.

Examples

```
# load example data
testData <- january2018

# add concatenated address variable
testData <- cs_address(testData, address = ileads_address, street = ileads_street, newVar = address)
```

`cs_collapse` *Collapse Months in Year List Object into Single Tibble*

Description

This function takes a year-list containing individual tibbles - one per month - that have been validated and collapses them into a single tibble.

Usage

```
cs_collapse(.data)
```

Arguments

`.data`	A list containing monthly crime data

Details

`cs_collapse` applies common sense variable classes to a number of variables. This is motivated by issues that originate with SLMPD `.csv` files. When they are imported, the **readr** package sometimes applies the incorrect variable classes because of formatting issues in the tables. Since the tables have inconsistent variable names and numbers of variables, all variables are imported as `chr` data. During `cs_collapse`'s execution, the following changes are made:

Count Converted to `int`

Crime Converted to `int`

District Converted to `int`

ILEADSAddress Converted to `int`

Neighborhood Converted to `int`

CADAddress Converted to `int`

XCoord Converted to `dbl`

YCoord Converted to `dbl`

Value

A tibble containing all crime data in a given year-list object.

Examples

```
# load example year-list object
load(system.file("testdata", "yearList17.rda", package = "compstatr", mustWork = TRUE))

# validate
cs_validate(yearList17, year = 2017)

# standaridze May, which has 26 variables
yearList17 <- cs_standardize(yearList17, month = "May", config = 26)

# validate again to confirm fix
cs_validate(yearList17, year = 2017)

# collapse now that the data are valid
crimeReports17 <- cs_collapse(yearList17)
```

---

cs_combine *Ensure Objects Contain Data Only For a Given Year*

Description

Since crimes are sometimes reported well after they are committed, objects created with `cs_collapse` often contain crimes that occurred in prior years. The `cs_combine` function ensures that objects contain only data for a given year, with the ability to add in crimes reported for the given year in later years.

Usage

```
cs_combine(type = "year", date, ...)
```

Arguments

`type`	"year" is the only valid input currently; year to date functionality is planned for a later update
`date`	For `type = "year"`, this should be the year of data to be returned. For `type = "ytd"`, this should be the last month to be included in each estimate.
`...`	An unquoted list of objects

Details

When applied to a single year's worth of data, `cs_combine` will subset out any crimes that occured in a year other than the one given for the `date` argument.

When applied to a range of objects, such as objects for 2017 and 2018, each object will be subset to identify crimes that occured in the year given for the `date` argument. This creates a more complete accounting of crime in a given year since it adds in crimes reported in subsequent years to the object. At the same time, crimes that occured prior to the given year will also be subset out to ensure the resulting object only contains crimes that occured in that given year.

Value

A tibble containing a selection of combined crime data for a given time period.

Examples

```
# load example year-list objects
load(system.file("testdata", "yearList17.rda", package = "compstatr", mustWork = TRUE))
load(system.file("testdata", "yearList18.rda", package = "compstatr", mustWork = TRUE))

# validate
cs_validate(yearList17, year = 2017)
cs_validate(yearList18, year = 2018)

# standaridze May for the 2017 object, which has 26 variables
yearList17 <- cs_standardize(yearList17, month = "May", config = 26)

# validate again to confirm fix
cs_validate(yearList17, year = 2017)

# collapse now that the data are valid
crimeReports17 <- cs_collapse(yearList17)
crimeReports18 <- cs_collapse(yearList18)

# combine to add all sample 2017 crimes reported in 2018 to a single 2017 object
# and remove from our 2017 object all sample crimes reported in 2017 that occured prior
# to that year
crime17 <- cs_combine(type = "year", date = 2017, crimeReports17, crimeReports18)
```

`cs_create_index` *Create Index of Available Months*

Description

Constructs a table for finding a given table of crime data or a set of tables (such as year to date or full year). This is largely needed for internal use when downloading tables, but is exported for reference and troubleshooting.

Usage

```
cs_create_index()
```

Value

A tibble with all available monthly crime tables, the `iframe` page they appear on, and their row number.

Examples

```
# create index
i <- cs_create_index()

# preview of index object
i
```

`cs_crime` *Identify Crimes*

Description

`cs_crime` can be used to easily identify crimes based on a specific single UCR categories or common groupings. This can be used on any police department's data where codes like `31111` (robbery with a firearm) or `142320` (malicious destruction of property) are used to identify crimes.

Usage

```
cs_crime(.data, var, newVar, crime)
```

Arguments

`.data`	A tibble or data frame
`var`	Name of variable with 5 or 6 digit crime codes
`newVar`	Name of output variable to be created with logical data
`crime`	A string describing the crime type to be identified

Details

The categories used here are derived from the U.S. Federal Bureau of Investigation's Uniform Crime Reporting codes. Valid inputs for the `crime` argument are as follows:

`"violent"` Violent crimes (homicide, rape, aggravated assault, and robbery)

`"property"` Property crimes (burglary, larceny, larceny of a motor vehicle, and arson)

`"part 1"` All violent and property crimes

`"homicide"` `"murder"` is also acceptable as input as is UCR code `1`

"rape" "forcible rape" is also acceptable as input as is UCR code 2

"robbery" UCR code 3 is also acceptable input

"agg assualt" "aggravated assualt" is also acceptable as input as is UCR code 4

"burglary" UCR code 5 is also acceptable input

"larceny-theft" "larceny" and "theft" are also acceptable inputs as is UCR code 6

"mv theft" "motor vehicle theft", "motor vehicle larceny", and "mv larceny" are also acceptable inputs as input as is UCR code 7

"arson" UCR code 8 is also acceptable input

"part 2" All other crimes

"assault" "other assaults" is also acceptable input as is UCR code 9

"forgery" "forgery and counterfeiting" is also acceptable input as is UCR code 10

"fraud" UCR code 11 is also acceptable input

"embezzlement" UCR code 12 is also acceptable input

"stolen prop" "stolen property" is also acceptable input as is UCR code 13

"vandalism" UCR code 14 is also acceptable input

"weapons" UCR code 15 is also acceptable input

"prostitution" "prostitution and commercialized vice" is also acceptable input as is UCR code 16

"sex offenses" UCR code 17 is also acceptable input

"drugs" "drug abuse violations" is also acceptable input as is UCR code 18

"gambling" UCR code 19 is also acceptable input

"family" "offenses against the family and children" is also acceptable input as is UCR code 20

"dwi" "driving under the influence" is also acceptable input as is UCR code 21

"liquor laws" UCR code 22 is also acceptable input

"drunkenness" UCR code 23 is also acceptable input

"discon" "disorderly conduct" is also acceptable input as is UCR code 24

"vagrancy" UCR code 25 is also acceptable input

"other" "all other offenses" is also acceptable input as is UCR code 26

"suspicion" UCR code 27 is also acceptable input

"curfew" "curfew and loitering laws-persons under 18" is also acceptable input as is UCR code 28

"runaway" "runaways-persons under 18" is also acceptable input as is UCR code 29

Value

A copy of the object with a logical vector that is TRUE if the given crime matches the category given in the function.

Examples

```
# load example data
testData <- january2018

# add logical vector for violent crimes
testData <- cs_crime(testData, var = crime, newVar = violentCrimes, crime = "violent")
```

`cs_crime_cat` *Categorize Crime*

Description

The SLMPD data contains 5 or 6 digit codes to refer to specific categories of crime. `cs_crime_cat` transforms these into either string, factor, or simplified numeric categories like "murder" or "aggravated assault". This can be used on any police department's data where codes like `31111` (robbery with a firearm) or `142320` (malicious destruction of property) are used to identify crimes.

Usage

```
cs_crime_cat(.data, var, newVar, output)
```

Arguments

`.data`	A tibble or data frame
`var`	Name of variable with 5 or 6 digit crime codes
`newVar`	Name of output variable to be created with simplified categories
`output`	Type of output - either `"string"`, `"factor"`, or `"numeric"`. If `"numeric"` is selected, the general UCR code will be returned (i.e. `1` for homicide, `3` for aggravated assault, etc.). Factor output will be returned in order of descending UCR code (i.e. beginning with homicide, which has a UCR code of `1`).

Details

The categories used here are derived from the U.S. Federal Bureau of Investigation's Uniform Crime Reporting codes.

Value

A copy of the object with the new output variable appended to it.

Examples

```
# load example data
testData <- january2018

# apply categories
testData <- cs_crime_cat(testData,var = crime, newVar = crimeCat, output = "numeric")

# preview categories
table(testData$crimeCat)

# apply categories
testData <- cs_crime_cat(testData,var = crime, newVar = crimeCat, output = "factor")

# preview categories
table(testData$crimeCat)

# apply categories
testData <- cs_crime_cat(testData,var = crime, newVar = crimeCat, output = "string")

# preview categories
table(testData$crimeCat)
```

---

cs_example *Load Example Files*

Description

Adds a sample set of twelve files, one for each month of 2017, to the specified path. These are not full data files; each file has twenty observations. They can be used to practice preparing, loading, standardizing, and collapsing data.

Usage

```
cs_example(path, overwrite = FALSE)
```

Arguments

path	File path where example data should be placed
overwrite	Overwrite files if they exist. If this is FALSE and the file exists an error will be thrown.

Examples

```
# create temporary directory
tmpdir <- tempdir()
fs::dir_create(paste0(tmpdir,"/data/"))

# load sample files into temporary directory
```

```
cs_example(path = paste0(tmpdir,"/data/"))

# list files
list.files(paste0(tmpdir,"/data/"))

# delete data
fs::dir_delete(paste0(tmpdir,"/data/"))
```

`cs_extract_month` *Extract Month from Year List Object*

Description

This function extracts a given month from a list containing 12 tibbles - one per month - for additional data cleaning prior to collapsing the list. Since months are ordered alphabetically in the year list objects, this function makes the process of extracting a particular month more intuitive.

Usage

```
cs_extract_month(.data, month)
```

Arguments

`.data`	A list containing monthly crime data
`month`	A string name or abbreviation of a month, or its numeric value. Acceptable inputs include, for example, "January", "january", "Jan", "jan", and 1.

Value

A tibble containing a single month worth of crime data.

See Also

`cs_replace_month`

Examples

```
# load example year-list object
load(system.file("testdata", "yearList17.rda", package = "compstatr", mustWork = TRUE))

# extract May
may17 <- cs_extract_month(yearList17, month = 5)
may17 <- cs_extract_month(yearList17, month = "May")
```

cs_filter_count *Remove Negative Counts*

Description

Removes the row that contains -1 in a specified column, indicating that the charge described in that observation has either been deemed unfounded or has been up-coded. For example, a victim of an aggravated assault dies, and the charge is changed to homicide.

Usage

```
cs_filter_count(.data, var)
```

Arguments

.data	A tibble or data frame
var	the name of the column

Value

A subset object with rows containing -1 removed

Examples

```
# load example data
testData <- january2018

# subset data to remove negative counts
testData <- cs_filter_count(testData, var = count)
```

cs_filter_crime *Filter Crimes*

Description

cs_filter_crime can be used to subset based on specific single UCR categories or common groupings. This can be used on any police department's data where codes like 31111 (robbery with a firearm) or 142320 (malicious destruction of property) are used to identify crimes.

Usage

```
cs_filter_crime(.data, var, crime)
```

Arguments

.data	A tibble or data frame
var	Name of variable with 5 or 6 digit crime codes
crime	A string describing the crime type to be identified

Details

The categories used here are derived from the U.S. Federal Bureau of Investigation's Uniform Crime Reporting codes. Valid inputs for the crime argument are as follows:

"violent" Violent crimes (homicide, rape, aggravated assault, and robbery)

"property" Property crimes (burglary, larceny, larceny of a motor vehicle, and arson)

"part 1" All violent and property crimes

"homicide" "murder" is also acceptable as input as is UCR code 1

"rape" "forcible rape" is also acceptable as input as is UCR code 2

"robbery" UCR code 3 is also acceptable input

"agg assualt" "aggravated assualt" is also acceptable as input as is UCR code 4

"burglary" UCR code 5 is also acceptable input

"larceny-theft" "larceny" and "theft" are also acceptable inputs as is UCR code 6

"mv theft" "motor vehicle theft", "motor vehicle larceny", and "mv larceny" are also acceptable inputs as input as is UCR code 7

"arson" UCR code 8 is also acceptable input

"part 2" All other crimes

"assault" "other assaults" is also acceptable input as is UCR code 9

"forgery" "forgery and counterfeiting" is also acceptable input as is UCR code 10

"fraud" UCR code 11 is also acceptable input

"embezzlement" UCR code 12 is also acceptable input

"stolen prop" "stolen property" is also acceptable input as is UCR code 13

"vandalism" UCR code 14 is also acceptable input

"weapons" UCR code 15 is also acceptable input

"prostitution" "prostitution and commercialized vice" is also acceptable input as is UCR code 16

"sex offenses" UCR code 17 is also acceptable input

"drugs" "drug abuse violations" is also acceptable input as is UCR code 18

"gambling" UCR code 19 is also acceptable input

"family" "offenses against the family and children" is also acceptable input as is UCR code 20

"dwi" "driving under the influence" is also acceptable input as is UCR code 21

"liquor laws" UCR code 22 is also acceptable input

"drunkenness" UCR code 23 is also acceptable input

"discon" "disorderly conduct" is also acceptable input as is UCR code 24

"vagrancy" UCR code 25 is also acceptable input

"other" "all other offenses" is also acceptable input as is UCR code 26

"suspicion" UCR code 27 is also acceptable input

"curfew" "curfew and loitering laws-persons under 18" is also acceptable input as is UCR code 28

"runaway" "runaways-persons under 18" is also acceptable input as is UCR code 29

Value

A subset object with only the specified crimes

Examples

```
# load example data
testData <- january2018

# subset data to retain only violent crimes
testData <- cs_filter_crime(testData, var = crime, crime = "violent")
```

---

cs_get_data *Download Crime Data from SLMPD*

Description

Downloads crime data from the SLMPD website.

Usage

```
cs_get_data(year, month, index)
```

Arguments

year	A year value in the style YYYY
month	Optional; a month number, name, or abbreviation - 1, "Jan", and "January" are all acceptible inputs.
index	Optional; an index object created with cs_create_index. Building the index prior to downloading data, especially if you are downloading multiple years worth of data, will result in dramatically faster execution times for this function.

Value

A year-list object ready for validation.

Examples

```
# create index
i <- cs_create_index()

# download single month
may18 <- cs_get_data(year = 2018, month = "May", index = i)

# preview single month
may18

# download full year
yearList18 <- cs_get_data(year = 2018, index = i)

# preview year list object
yearList18
```

`cs_last_update` *Date of Last Crime Data Update from SLMPD*

Description

Data are updated by SLMPD on their crime statistics site on a monthly basis. This function returns the date of the last update.

Usage

```
cs_last_update(output = "string")
```

Arguments

`output`	A character scalar; if `"string"` the date will be returned in the style of `"January 2019"`. If `"date"` the date will be returned as a YYYY-MM-DD date object.

Value

The date of the last posted data set in the format specified in the `output` parameter.

Examples

```
# obtain data of last update
cs_last_update()
```

cs_load_year *Create Year List Object*

Description

cs_load_year is used to load a set of .csv files contained in the given directory. This should be used to load a full year worth of data or a partial year. There should be no more than 12 files in a given path, and all should correspond to the same year. All columns will be read in as character data in order to address inconsistencies in how the data are created. When cs_collapse is executed, variables will be converted numeric when doing so is applicable.

Usage

```
cs_load_year(path)
```

Arguments

path	A file path

Value

A year-list object containing 12 tibbles - one per month - worth of crime data stored within a list.

Examples

```
# create temporary directory
tmpdir <- tempdir()
fs::dir_create(paste0(tmpdir,"/data/"))

# load sample files into temporary directory
cs_example(path = paste0(tmpdir,"/data/"))

# prep sample files
cs_prep_year(path = paste0(tmpdir,"/data/"))

# load sample files
yearList17 <- cs_load_year(path = paste0(tmpdir,"/data/"))

# delete data
fs::dir_delete(paste0(tmpdir,"/data/"))

# print year-list object
yearList17
```

cs_missingXY *Identify Missing Coordinates*

Description

cs_missingXY compares X and Y coordinates and adds a logical column that identifies observations that are missing coordinate data.

Usage

```
cs_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 tibble or data frame with a logical vector appended to it.

Examples

```
# load example data
testData <- january2018

# identify missing x and y coordinates
testData <- cs_missingXY(testData, varX = x_coord, varY = y_coord, newVar = missingXY)
```

cs_parse_date *Separate Date Occur*

Description

Creates two columns. One contains month, day, and year and the other contains hour, and minute.

Usage

```
cs_parse_date(.data, var, dateVar, timeVar, tz = NULL, keepDateTime = TRUE)
```

Arguments

.data	A tibble or data frame
var	A column containing month, day, year, and time separated by /
dateVar	Name of new column to contain date data
timeVar	Name of new column to contain time data
tz	String name of timezone, defaults to system's timezone
keepDateTime	A logical scalar. Keep an intermediate dateTime variable if TRUE.

Value

A copy of the object with two columns appended. One is the time data and the other is the date data.

Examples

```
# load example data
testData <- january2018

# parse date occured
testData <- cs_parse_date(testData, var = date_occur, dateVar = dateOcc, timeVar = timeOcc)
```

---

cs_parse_month *Separate Coded Month*

Description

Separates a column containing coded year and coded month separated by "-" into two columns and removes the input column.

Usage

```
cs_parse_month(.data, var, yearVar, monthVar)
```

Arguments

.data	A tibble or data frame
var	the variable containing coded month and coded year
yearVar	the name of the column to contain the year data
monthVar	the name of the column to contain month data

Value

Returns a copy of the object with two new columns for the coded year and coded month appended to it.

Examples

```
# load example data
testData <- january2018

# parse CodedMonth
testData <- cs_parse_month(testData, var = coded_month, yearVar = reportYear,
    monthVar = reportMonth)
```

`cs_prep_year` *Prepare Raw Data*

Description

Data downloaded from the St. Louis Metropolitan Police Department are downloaded with incorrect file paths - e.g. `January2008.CSV.html`. This function iterates over all files in a given path and replaces their file extensions. Thus `January2008.CSV.html` will be replaced by `january2008.csv`. There should be no more than 12 files in a given path, and all should correspond to the same year.

Usage

```
cs_prep_year(path, verbose = FALSE)
```

Arguments

`path`	File path where raw STLMPD data are
`verbose`	If `TRUE`, returns a tibble with results; otherwise if `FALSE`, no output is returned.

Value

A tibble containing old file names and new file names for reference is `verbose = TRUE`. Otherwise, no output is returned. This function will change all problematic filenames in the specified path.

Examples

```
# create temporary directory
tmpdir <- tempdir()
fs::dir_create(paste0(tmpdir,"/data/"))

# load sample files into temporary directory
cs_example(path = paste0(tmpdir,"/data/"))

# list files
list.files(paste0(tmpdir,"/data/"))

# prep sample files
cs_prep_year(path = paste0(tmpdir,"/data/"))
```

```
# list files again
list.files(paste0(tmpdir,"/data/"))

# delete data
fs::dir_delete(paste0(tmpdir,"/data/"))

# create temporary directory
fs::dir_create(paste0(tmpdir,"/data/"))

# load sample files into temporary directory
cs_example(path = paste0(tmpdir,"/data/"))

# prep sample files
cs_prep_year(path = paste0(tmpdir,"/data/"), verbose = TRUE)

# delete data again
fs::dir_delete(paste0(tmpdir,"/data/"))
```

---

cs_projectXY *Project Data*

Description

cs_projectXY converts STLMPD data into a simple features object using the XCoord and YCoord columns.

Usage

```
cs_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	integer with the EPSG code, or character with proj4string representing the coordinate reference system

Value

A sf object with the crime data projected for mapping.

Examples

```
# load example data
testData <- january2018

# identify missing x and y coordinates
testData <- cs_missingXY(testData, varX = x_coord, varY = y_coord, newVar = missingXY)

# subset to remove missing data
testData <- dplyr::filter(testData, missingXY == FALSE)

# project data
testData_sf <- cs_projectXY(testData, varX = x_coord, varY = y_coord)

# project data and transform to new CRS
testData_sf <- cs_projectXY(testData, varX = x_coord, varY = y_coord, crs = 4269)
```

`cs_replace0` *Replace Coordinates with 0 Value with NA*

Description

This function a specified column from the data frame and replaces cells that have the value 0 with NA

Usage

```
cs_replace0(.data, var)
```

Arguments

`.data`	A tibble or data frame
`var`	Name of column containing coordinate data

Value

A tibble or data frame with the coordinate column updated.

Examples

```
# load example data
testData <- january2018

# replace 0s in the x and y coordinate variables
testData <- cs_replace0(testData, var = x_coord)
testData <- cs_replace0(testData, var = y_coord)
```

cs_replace_month *Extract Month from Year-list Object*

Description

This function replaces a single month worth of crime data that has previously been extracted from a year-list object.

Usage

```
cs_replace_month(.data, month, monthData)
```

Arguments

.data	A year list object
month	A string name or abbreviation of a month, or its numeric value. Acceptable inputs include, for example, "January", "january", "Jan", "jan", and 1.
monthData	A tibble containing a single month worth of crime data.

Value

An updated year-list object.

See Also

cs_extract_month

Examples

```
# load example year-list object
load(system.file("testdata", "yearList17.rda", package = "compstatr", mustWork = TRUE))

# extract May
may17 <- cs_extract_month(yearList17, month = 5)

# replace
yearList17 <- cs_replace_month(yearList17, month = 5, monthData = may17)
yearList17 <- cs_replace_month(yearList17, month = "May", monthData = may17)
```

cs_standardize *Standardized Variables*

Description

Different time points of SLMPD have different numbers of variables and different names for those variables that are included in both sets of releases. This function reformats non-standard configurations to a 20 variable standard.

Usage

```
cs_standardize(.data, month, config = 18)
```

Arguments

.data	A tbl
month	An option string name or abbreviation of a month, or its numeric value. Acceptable inputs include, for example, "January", "january", "Jan", "jan", and 1. If all months in a year-list need to be standardized (this is applicable, as of March 2019, to all years from 2008 through 2012), the month should be given as "all" to standardize them en masse.
config	The non-standard configuration, either 18 or 26

Details

For all months prior to 2013 and approximately half of the months during 2013, SLMPD data are released with 18 variables. For one month, May 2017, the data are released with 26 variables. This function can be used to either edit an entire year list object or to edit only a specified month within it. In general, years 2008 through 2012 should be edited en masse while the month specification can be used to edit the months in 2013 and 2017 that are non-standard.

Examples

```
# load example year-list object
load(system.file("testdata", "yearList17.rda", package = "compstatr", mustWork = TRUE))

# validate
cs_validate(yearList17, year = 2017)

# standaridze May, which has 26 variables
yearList17 <- cs_standardize(yearList17, month = "May", config = 26)

# validate again to confirm fix
cs_validate(yearList17, year = 2017)
```

`cs_validate` *Validate Year List Object*

Description

Data from SLMPD are released with a number of problems that `cs_validate` is designed to identify.

Usage

```
cs_validate(.data, year, verbose = FALSE)
```

Arguments

`.data`	A tbl
`year`	A string representing the year being checked, e.g. `"2008"`
`verbose`	A logical scalar. If `TRUE`, a full validation report summarizing results will be returned. If `FALSE`, a single value will be returned.

Details

`cs_validate` performs a total of five checks on the given year-list object. Each test is summarized in the `verbose = TRUE` output:

valMonth Each tibble within a year-list is named for the month it represents. Does the named month match the month that the data represent?

valYear Does the year provided for the `year` argument match the year that the data represent?

oneMonth Does each tibble represent only one month of data?

varCount Does each tibble have the correct number of variables (20)?

valVars Does each tibble have the correct variable names?

For all months prior to 2013 and approximately half of the months during 2013, SLMPD data are released with 18 variables. For one month, May 2017, the data are released with 26 variables. These problems are identified most easily by using `cs_validate`.

Value

A tibble with validation results.

Examples

```
# load example year-list object
load(system.file("testdata", "yearList17.rda", package = "compstatr", mustWork = TRUE))

# simple validation
cs_validate(yearList17, year = 2017)
```

```
# verbose validation
cs_validate(yearList17, year = 2017, verbose = TRUE)
```

`january2018` *Crimes in St. Louis, January 2018*

Description

A data set containing all reported crimes in St. Louis, Missouri during January 2018.

Usage

```
data(january2018)
```

Format

A tibble with 3825 rows and 20 variables:

complaint complaint record number

coded_month year and month crime reported

date_occur date and time of crime

flag_crime Returns a Y when a crime occurred

flag_unfounded Reported crime was investigated and determined to be unfounded

flag_administrative Reported crime had a change in classification

count Returns a 1 or -1 for counting purposes

flag_cleanup Returns information if administrative cleanup occured

crime Uniform Crime Reporting code

district Number corresponding to the police determined district

description Name of the crime

ileads_address I/Leads system address

ileads_street I/Leads system street

neighborhood Number corresponding to a neighborhood

location_name Common "Location Name" (i.e. Zoo, Scottrade Center, etc.)

location_comment Information to provide context to the location (i.e. Alley, Restaraunt Name)

cad_address The Computer-Aided Dispatch address is the reported address by the 911 caller

cad_street The Computer-Aided Dispatch street is the reported street by the 911 caller

x_coord X-coordinates in the NAD83 format

y_coord Y-coordinates in the NAD83 format

Source

St. Louis Metropolitan Police Department

Examples

```
str(january2018)
head(january2018)
```

Index