Package 'wildviz'

August 23, 2021

Type Package

Title Compiles and Visualizes Wildfire, Climate, and Air Quality Data

Version 0.1.2

Date 2021-08-21

Description Fetches data from three disparate data sources and allows user to perform analyses on them. It offers two core components: 1. A robust data retrieval and preparation infrastructure for wildfire, climate, and air quality index data and 2. A simple, informative, and interactive visualizations of the aforementioned datasets for California counties from 2011 through 2015. The sources of data are: wildfire data from Kaggle <https://www.kaggle.com/rtatman/188-million-us-wildfires>, climate data from the National Oceanic and Atmospheric Administration <https://www.ncdc.noaa.gov/cdo-web/token>, and air quality data from the Environmental Protection Agency <https://aqs.epa.gov/aqsweb/documents/data_api.html>.

URL https://github.com/bradraff/wildviz

BugReports https://github.com/bradraff/wildviz/issues

License GPL-2

LazyData true

Imports DBI, dplyr, httr, jsonlite, lubridate, purrr, rnoaa, RSQLite, shiny, tibble, tidyr, shinythemes, ggplot2, plotly, ggthemes

Encoding UTF-8

RoxygenNote 7.1.1

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown, markdown, testthat

VignetteBuilder knitr

LazyDataCompression xz

NeedsCompilation no

Author Bradley Rafferty [aut, cre], Daniel Chung [aut]

Maintainer Bradley Rafferty <brjrafferty@gmail.com>

Repository CRAN

Date/Publication 2021-08-23 07:50:02 UTC

R topics documented:

`aqi` *Air quality data records for California from 2011 through 2015*

Description

A dataframe containing Calfornia air quality records at the county level. It is created from the EPA's Air Quality System (AQS) API by fetching the AQI along with other atmospheric measurements like CO, Ozone, NO2, PM2.5, and PM10 and reforrmatting the tibble as a tidy data consisting of a row per county per day.

Usage

```
aqi
```

Format

A dataframe with 102752 rows and 10 variables:

state_code Two-digit code from the FIPS publication 6-4 for states

county_code Three-digit code from the FIPS publication 6-4 for counties

county_name County name from the FIPS publication 6-4 for counties

date Date on which the air quality measures were recorded

aqi AQI level

co Carbon monoxide, in Parts per million

ozone Ozone, in Parts per million

no2 Nitrogen dioxide (NO2), in Parts per billion

pm25 PM2.5, in Micrograms/cubic meter (LC)

pm10 PM10, in Micrograms/cubic meter (25 C)

Details

AQS contains ambient air sample data collected by state, local, tribal, and federal air pollution control agencies from thousands of monitors around the nation.

Source

https://aqs.epa.gov/aqsweb/documents/data_api.html

`ave_daily` *Average daily weather data across multiple stations.*

Description

Returns a dataframe with daily weather averaged across stations, as well as columns showing the number of stations contributing to the average for each variable and each day.

Usage

```
ave_daily(weather_data)
```

Arguments

`weather_data`	A dataframe with daily weather observations. This dataframe is returned from the `rnoaa` function `meteo_pull_monitors`.

Value

A dataframe of daily weather averaged across weather stations

`climate` *Daily weather summaries for California from 2011 through 2015*

Description

A dataframe containing daily weather summaries for California at the county level. It is created from summarizing the measurements from GHCND stations to the daily level and by filtering on the date (2011 - 2015), state (CA), and key cols.

Usage

```
climate
```

Format

A dataframe with 105850 rows and 7 variables:

fips Five-digit code combining the state and county codes from the FIPS publication 6-4 for counties

date Date on which the climate measures were recorded

prcp precipitation, in mm

snow snowfall, in mm

snwd snow depth, in mm

tmax maximum temperature, in degrees Celsius

tmin minumum temperature, in degrees Celsius

Details

Data from the Daily Global Historical Climatology Network (GHCN-Daily) through the NOAA FTP server. The data is archived at the National Centers for Environmental Information (NCEI) (formerly the National Climatic Data Center (NCDC)), and spans from the 1800s to the current year.

Source

https://www1.ncdc.noaa.gov/pub/data/cdo/documentation/GHCND_documentation.pdf

create_wildfire *Create US wildfires dataframe from the Kaggle US Wildfire SQLite db*

Description

Create US wildfires dataframe from the Kaggle US Wildfire SQLite db

Usage

```
create_wildfire(
  db_name,
  state_abbrev = NULL,
 cols = c("FIRE_NAME", "DISCOVERY_DATE", "CONT_DATE", "STAT_CAUSE_DESCR", "FIRE_SIZE",
  "FIRE_SIZE_CLASS", "LATITUDE", "LONGITUDE", "STATE", "FIPS_CODE", "FIPS_NAME"),
  year_min = 1992,
  year_max = 2015
)
```

Arguments

db_name	File path of the SQLite wildfire database.
state_abbrev	Abbreviations of the states to retrieve the wildfire data for.
cols	Columns to select from the wildfire database. For more info: (https://www.kaggle.com/rtatman/188-million-us-wildfires)
year_min	earliest year to pull the air quality data for, starts January 1st.
year_max	latest year to pull the air quality data for, ends December 31st.

Value

A dataframe of wildfires that occurred in the United States.

Note

The function relies on the SQLite db available on Kaggle: (https://www.kaggle.com/rtatman/188-million-us-wildfires) Unfortunately, Kaggle API does not support R at this time. Download the '188-million-us-wildfires.zip' file, and provide the path to the function as db_name, i.e. db_name = 'data-raw/FPA_FOD_20170508.sqlite'

Examples

```
## Not run:
fires <- create_wildfire(db_name = 'data-raw/FPA_FOD_20170508.sqlite',
                         state_abbrev = c('CA', 'NY'),
                         cols=c('FIRE_NAME', 'DISCOVERY_DATE', 'CONT_DATE',
                                'STAT_CAUSE_DESCR', 'FIRE_SIZE', 'FIRE_SIZE_CLASS',
                            'LATITUDE', 'LONGITUDE', 'STATE', 'FIPS_CODE', 'FIPS_NAME'),
                         year_min = 1992,
```

```
                          year_max = 2015)

## End(Not run)
```

`daily_aqi` *Retrieve the air quality data.*

Description

Retrieve the air quality data.

Usage

```
daily_aqi(
  aqs_api_email,
  aqs_api_key,
  fips_list,
  metric_list = c(81102, 88101, 42101, 44201, 42602),
  year_min = 2015,
  year_max = 2015
)
```

Arguments

`aqs_api_email`	AQS API email
`aqs_api_key`	AQS API key
`fips_list`	A vector of FIPS codes per county. One can use the output from the `get_counties` function as follows: `fips_list=pull(counties,'fips')`
`metric_list`	list of codes that represent different air quality metrics 81101 - PM10, 88101 - PM2.5, 42101 - CO, 42602 - NO2, 44201 - Ozone.
`year_min`	earliest year to pull the air quality data for, starts January 1st.
`year_max`	latest year to pull the air quality data for, ends December 31st.

Value

A dataframe of daily AQI data, including the metrics specified.

Note

The function uses the AQS API to fetch the data from the API endpoints. Use the following service to register as a user: A verification email will be sent to the email account specified. To register using the email address create and request this link (Replace `<myemail@example.com>` in the example with your email address.): (https://aqs.epa.gov/data/api/signup?email=myemail@example.com) You then need to set the email and the key in the .Renviron file as follows, i.e. aqs_api_email=`<myemail@example.com>` aqs_api_key=testkey1234

AQS AQI has request limits and ToS: (https://aqs.epa.gov/aqsweb/documents/data_api.html#terms). The function intentionally adds a 10 second delay between each call (per year, per county) but pay attention to the limits as the account may be disabled. It is also recommended to process no more than five years at a time, as the API request occasionally times out and data may be lost.

Examples

```
## Not run:
aqs_api_email = Sys.getenv("aqs_api_email")
aqs_api_key = Sys.getenv("aqs_api_key")

state_codes <- get_state_code(aqs_api_email = aqs_api_email,
                              aqs_api_key = aqs_api_key,
                              state_names = c('California'))
counties <- get_counties(aqs_api_email = aqs_api_email,
                         aqs_api_key = aqs_api_key,
                         state_codes = state_codes)
aqi <- daily_aqi(aqs_api_email = aqs_api_email,
                 aqs_api_key = aqs_api_key,
                 fips_list = counties$fips,
                 year_min = 2015,
                 year_max = 2015)

## End(Not run)
```

---

daily_climate *Retrieve average daily weather data per US county.*

Description

Retrieve average daily weather data per US county.

Usage

```
daily_climate(
  fips,
  var = c("prcp", "snow", "snwd", "tmax", "tmin"),
  date_min = "2015-01-01",
  date_max = "2015-12-31",
  coverage = 0.9
)
```

Arguments

fips	A string with the five-digit U.S. FIPS code of a county in numeric, character, or factor format.

var — A character vector specifying desired weather variables. For example, `var = c("tmin","tmax","prcp")` for maximum temperature, minimum temperature, and precipitation. The default is `"all"`, which includes all available weather variables at any weather station in the county. For a full list of all possible variable names, see NOAA's README file for the Daily Global Historical Climatology Network (GHCN-Daily) at https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/readme.txt. Many of the weather variables are available for some, but not all, monitors, so your output from this function may not include all the variables specified using this argument. If you specify a variable here but it is not included in the output dataset, it means that it was not available in the time range for any monitor in the county.

date_min — A string with the desired starting date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates including and after the specified date.

date_max — A string with the desired ending date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates up to and including the specified date.

coverage — A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors. The default is to include all monitors with any available data (i.e., `coverage = 0`).)

Value

A dataframe of daily weather data averaged across multiple stations.

Note

The function uses the NOAA API to identify the weather monitors within a U.S. county, you will need to get an access token from NOAA to use this function. Visit NOAA's token request page (https://www.ncdc.noaa.gov/cdo-web/token) to request a token by email. You then need to set that API code in your R session (e.g., using `options(noaakey = "your key")`, replacing "your key" with the API key you've requested from NOAA).

Examples

```
## Not run: s
climate <- daily_climate(fips = '06001',
                         var = c('prcp', 'snow', 'snwd', 'tmax', 'tmin'),
                         date_min = '2015-01-01',
                         date_max = '2015-12-31',
                         coverage = 0.90)

## End(Not run)
```

daily_climate_counties *Retrieve average daily weather data for a list of US counties.*

Description

Retrieve average daily weather data for a list of US counties.

Usage

```
daily_climate_counties(
  fips_list,
  var = c("prcp", "snow", "snwd", "tmax", "tmin"),
  date_min = "2015-01-01",
  date_max = "2015-12-31",
  coverage = 0.9
)
```

Arguments

fips_list	A vector of FIPS codes of the counties to pull the daily weather data for. One can use the output from the `get_counties` function as follows: `fips_list=pull(counties,'fips')`
var	A character vector specifying desired weather variables. For example, `var = c("tmin","tmax","prcp")` for maximum temperature, minimum temperature, and precipitation. The default is `"all"`, which includes all available weather variables at any weather station in the county. For a full list of all possible variable names, see NOAA's README file for the Daily Global Historical Climatology Network (GHCN-Daily) at https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/readme.txt. Many of the weather variables are available for some, but not all, monitors, so your output from this function may not include all the variables specified using this argument. If you specify a variable here but it is not included in the output dataset, it means that it was not available in the time range for any monitor in the county.
date_min	A string with the desired starting date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates including and after the specified date.
date_max	A string with the desired ending date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates up to and including the specified date.
coverage	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors. The default is to include all monitors with any available data (i.e., `coverage = 0`).)

Value

A dataframe of daily weather data averaged across multiple stations for each county in list.

Note

The function uses the NOAA API to identify the weather monitors within a U.S. county, you will need to get an access token from NOAA to use this function. Visit NOAA's token request page (https://www.ncdc.noaa.gov/cdo-web/token) to request a token by email. You then need to set that API code in your R session (e.g., using `options(noaakey = "your key")`, replacing "your key" with the API key you've requested from NOAA).

Examples

```
## Not run:
climate <- daily_climate_counties(fips_list = c('06001', '06003'),
                                  var = c('prcp', 'snow', 'snwd', 'tmax', 'tmin'),
                                  date_min = '2015-01-01',
                                  date_max = '2015-12-31',
                                  coverage = 0.90)

## End(Not run)
```

---

daily_df *Return average daily weather data for a particular county.*

Description

Returns a list with data on weather and stations for a selected county. This function serves as a wrapper to several functions from the `rnoaa` package, which pull weather data from all relevant stations in a county. This function filters and averages data returned by `rnoaa` functions across all weather stations in a county based on user-specified coverage specifications.

Usage

```
daily_df(
  stations,
  coverage = NULL,
  var = "all",
  date_min = NULL,
  date_max = NULL,
  average_data = TRUE
)
```

Arguments

stations	A dataframe containing station metadata, returned from the function daily_stations.
coverage	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors. The default is to include all monitors with any available data (i.e., coverage = 0).)
var	A character vector specifying desired weather variables. For example, var = c("tmin","tmax","prcp") for maximum temperature, minimum temperature, and precipitation. The default is "all", which includes all available weather variables at any weather station in the county. For a full list of all possible variable names, see NOAA's README file for the Daily Global Historical Climatology Network (GHCN-Daily) at https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/readme.txt. Many of the weather variables are available for some, but not all, monitors, so your output from this function may not include all the variables specified using this argument. If you specify a variable here but it is not included in the output dataset, it means that it was not available in the time range for any monitor in the county.
date_min	A string with the desired starting date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates including and after the specified date.
date_max	A string with the desired ending date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates up to and including the specified date.
average_data	TRUE / FALSE to indicate if you want the function to average daily weather data across multiple monitors. If you choose FALSE, the function will return a dataframe with separate entries for each monitor, while TRUE (the default) outputs a single estimate for each day in the dataset, giving the average value of the weather metric across all available monitors in the county that day.

Value

A list with two elements. daily_data is a dataframe of daily weather data averaged across multiple monitors and includes columns ("var"_reporting) for each weather variable showing the number of stations contributing to the average for that variable on that day. The element station_df is a dataframe of station metadata for each station contributing weather data. A weather station will have one row per weather variable to which it contributes data. In addition to information such as station id, name, latitude, and longitude, the station_df dataframe includes statistical information about weather values contributed by each station for each weather variable. These statistics include calc_coverage (the percent of non-missing values for each station-weather variable combination for the specified date range), standard_dev (standard deviation), max, and min, (giving the minimum and maximum values), and range, giving the range of values in each station-weather variable combination. The element radius is the calculated radius within which stations were pulled from the county's center. Elements lat_center and lon_center are the latitude and longitude of the county's center.

Note

Because this function uses the NOAA API to identify the weather monitors within a U.S. county, you will need to get an access token from NOAA to use this function. Visit NOAA's token request page (https://www.ncdc.noaa.gov/cdo-web/token) to request a token by email. You then need to set that API code in your R session (e.g., using `options(noaakey = "your key")`, replacing "your key" with the API key you've requested from NOAA). See the package vignette for more details.

Examples

```
## Not run:
stations <- daily_stations(fips = "12086", date_min = "2010-01-01",
                           date_max = "2010-02-01")
fips_list <- daily_df(stations = stations, coverage = 0.90,
                 var = c("tmax", "tmin", "prcp"),
                 date_min = "2010-01-01", date_max = "2010-02-01")
averaged_data <- fips_list$daily_data
head(averaged_data)
station_info <- fips_list$station_df
head(station_info)

## End(Not run)
```

---

daily_stations *NOAA NCDC station IDs per county.*

Description

Returns a dataframe with NOAA NCDC station IDs for a single U.S. county. This function has options to filter stations based on maximum and minimum dates, as well as percent data coverage.

Usage

```
daily_stations(fips, date_min = NULL, date_max = NULL)
```

Arguments

fips	A string with the five-digit U.S. FIPS code of a county in numeric, character, or factor format.
date_min	A string with the desired starting date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates including and after the specified date.
date_max	A string with the desired ending date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates up to and including the specified date.

Value

A dataframe with NOAA NCDC station IDs for a single U.S. county.

Note

Because this function uses the NOAA API to identify the weather monitors within a U.S. county, you will need to get an access token from NOAA to use this function. Visit NOAA's token request page (https://www.ncdc.noaa.gov/cdo-web/token) to request a token by email. You then need to set that API code in your R session (e.g., using `options(noaakey = "your key")`, replacing "your key" with the API key you've requested from NOAA). See the package vignette for more details.

Examples

```
## Not run:
stations_36005 <- daily_stations("36005")
stations_36005

miami_stations <- daily_stations("12086", date_min = "1999-01-01",
                                 date_max = "2012-12-31")
miami_stations

## End(Not run)
```

---

filter_coverage *Filter stations based on "coverage" requirements.*

Description

Filters available weather stations based on a specified required minimum coverage (i.e., percent non-missing daily observations). Weather stations with non-missing data for fewer days than specified by `coverage` will be excluded from the county average.

Usage

```
filter_coverage(coverage_df, coverage = 0)
```

Arguments

`coverage_df`	A dataframe as returned by the `meteo_coverage` function in the `rnoaa` package
`coverage`	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors).

Value

A dataframe with stations that meet the specified coverage requirements for weather variables included in the `coverage_df` dataframe passed to the function.

`get_counties` *Get list of county FIPS codes by providing the state codes, i.e. California = '06'.*

Description

Get list of county FIPS codes by providing the state codes, i.e. California = '06'.

Usage

```
get_counties(aqs_api_email, aqs_api_key, state_codes)
```

Arguments

`aqs_api_email`	AQS API email
`aqs_api_key`	AQS API key
`state_codes`	A vector of standard state codes according to the Federal Information Processing Standard Publication. This is an output of the `get_state_code` function.

Value

A dataframe of combined, 5 digit state + county FIPS codes ('fips') and their corresponding county names ('name').

Note

The function uses the AQS API to fetch the data from the API endpoints. Use the following service to register as a user: A verification email will be sent to the email account specified. To register using the email address create and request this link (Replace `<myemail@example.com>` in the example with your email address.): (https://aqs.epa.gov/data/api/signup?email=myemail@example.com) You then need to set the email and the key in the .Renviron file as follows, i.e. aqs_api_email=<myemail@example.com> aqs_api_key=testkey1234

AQS AQI has request limits and ToS: (https://aqs.epa.gov/aqsweb/documents/data_api.html#terms). The function intentionally adds a 10 second delay between each call (per year, per county) but pay attention to the limits as the account may be disabled.

Examples

```
## Not run:
aqs_api_email = Sys.getenv('aqs_api_email')
aqs_api_key = Sys.getenv('aqs_api_key')

state_codes <- get_state_code(aqs_api_email = aqs_api_email,
                              aqs_api_key = aqs_api_key,
                              state_names = c('California', 'New York'))
counties <- get_counties(aqs_api_email = aqs_api_email,
                         aqs_api_key = aqs_api_key,
                         state_codes = state_codes)

## End(Not run)
```

`get_state_code` *Get state code from the state names.*

Description

Get state code from the state names.

Usage

```
get_state_code(aqs_api_email, aqs_api_key, state_names)
```

Arguments

`aqs_api_email`	AQS API email
`aqs_api_key`	AQS API key
`state_names`	A vector containing state names.

Value

A vector of state code per state name provided.

Note

The function uses the AQS API to fetch the data from the API endpoints. Use the following service to register as a user: A verification email will be sent to the email account specified. To register using the email address create and request this link (Replace <myemail@example.com> in the example with your email address.): (https://aqs.epa.gov/data/api/signup?email=myemail@example.com) You then need to set the email and the key in the .Renviron file as follows, i.e. aqs_api_email=<myemail@example.com> aqs_api_key=testkey1234

Examples

```
## Not run:
aqs_api_email = Sys.getenv('aqs_api_email')
aqs_api_key = Sys.getenv('aqs_api_key')

state_codes <- get_state_code(aqs_api_email = aqs_api_email,
                              aqs_api_key = aqs_api_key,
                              state_names = c('California', 'New York'))

## End(Not run)
```

`master` *Master dataset combining the wildfires, AQI, and climate data for California from 2011 through 2015*

Description

The master dataframe takes each wildfire occurrence in California between 2011 through 2015, and joins the AQI and climate data for 30 days prior to the DISCOVERY_DATE (fire discovery date) and 30 days post CONT_DATE (fire contained date). The data is utilized by the Shiny app to create dashboards.

Usage

```
master
```

Format

A dataframe with 1105972 rows and 23 variables:

FIRE_NAME Name of the incident, from the fire report (primary) or ICS-209 report (secondary)

DISCOVER_DATE Date on which the fire was discovered or confirmed to exist

CONT_DATE Date on which the fire was declared contained or otherwise controlled

STAT_CAUSE_DESCR Description of the (statistical) cause of the fire

FIRE_SIZE Estimate of acres within the final perimeter of the fire

FIRE_SIZE_CLASS Code for fire size based on the number of acres within the final fire perimeter expenditures (A=greater than 0 but less than or equal to 0.25 acres, B=0.26-9.9 acres, C=10.0-99.9 acres, D=100-299 acres, E=300 to 999 acres, F=1000 to 4999 acres, and G=5000+ acres)

LATITUDE Latitude (NAD83) for point location of the fire (decimal degrees)

LONGITUDE Longitude (NAD83) for point location of the fire (decimal degrees)

STATE Two-letter alphabetic code for the state in which the fire burned

FIPS_NAME County name from the FIPS publication 6-4 for representation of counties

fips Five-digit code combining the state and county codes from the FIPS publication 6-4 for counties

clim_date Date on which the climate measures were recorded

prcp precipitation, in mm

snow snowfall, in mm

snwd snow depth, in mm

tmax maximum temperature, in degrees Celsius

tmin minumum temperature, in degrees Celsius

aqi AQI level

co Carbon monoxide, in Parts per million

ozone Ozone, in Parts per million

no2 Nitrogen dioxide (NO2), in Parts per billion

pm25 PM2.5, in Micrograms/cubic meter (LC)

pm10 PM10, in Micrograms/cubic meter (25 C)

`resetDefaults` *Function to reset defaults*

Description

Function to reset defaults

Usage

```
resetDefaults(
  db_name = "data-raw/FPA_FOD_20170508.sqlite",
  aqs_api_email = Sys.getenv("aqs_api_email"),
  aqs_api_key = Sys.getenv("aqs_api_key"),
  year_min = 2001,
  year_max = 2015
)
```

Arguments

`db_name`	file path for the wildfires SQLite database
`aqs_api_email`	API email for the AQS API
`aqs_api_key`	API key for the AQS API
`year_min`	minimum year to process for datasets
`year_max`	minimum year to process for datasets

Value

a list of default values invisibly

Examples

```
resetDefaults()
```

setDefaults *Set the default param names to values*

Description

Set the default param names to values

Usage

```
setDefaults(name, value)
```

Arguments

name	the parameter name
value	the value to assign

Value

the new list of defaults invisibly

Examples

```
setDefaults('year_min', 2001)
```

wildfires *Geo-referenced California wildfire records from 2011 through 2015*

Description

A dataframe containing Calfornia wildfire records at the county level. It is created from the Kaggle US wildfire records data by filtering on the date (2011- 2015), state (CA), and key cols and cleaning up the missing counties and reformatting the dates.

Usage

```
wildfires
```

Format

A dataframe with 31579 rows and 11 variables:

FIRE_NAME Name of the incident, from the fire report (primary) or ICS-209 report (secondary)

DISCOVER_DATE Date on which the fire was discovered or confirmed to exist

CONT_DATE Date on which the fire was declared contained or otherwise controlled

STAT_CAUSE_DESCR Description of the (statistical) cause of the fire

FIRE_SIZE Estimate of acres within the final perimeter of the fire

FIRE_SIZE_CLASS Code for fire size based on the number of acres within the final fire perimeter expenditures (A=greater than 0 but less than or equal to 0.25 acres, B=0.26-9.9 acres, C=10.0-99.9 acres, D=100-299 acres, E=300 to 999 acres, F=1000 to 4999 acres, and G=5000+ acres)

LATITUDE Latitude (NAD83) for point location of the fire (decimal degrees)

LONGITUDE Longitude (NAD83) for point location of the fire (decimal degrees)

STATE Two-letter alphabetic code for the state in which the fire burned

FIPS_CODE Three-digit code from the FIPS publication 6-4 for counties

FIPS_NAME County name from the FIPS publication 6-4 for counties

Details

This data publication contains a spatial database of wildfires that occurred in the United States from 1992 to 2015.

Source

https://www.kaggle.com/rtatman/188-million-us-wildfires

---

wildvizApp *Run the shiny app from the wildviz package*

Description

Runs the wildviz shiny app for data visualization and exploration. Runs on the master, aqi, wildfires, and climate datasets provided within the wildviz package, which are specific to California and whose dates range from 2011 through 2015.

Usage

```
wildvizApp()
```

Value

No return value, called for side effects

Index