Package 'purexposure'

August 26, 2018

Title Pull and Calculate Exposure to CA Pesticide Use Registry Records

Version 0.1.0

Description Pulls and cleans California Pesticide Use Report (PUR) data sets to visualize and calculate application of and exposure to active ingredients present in applied pesticides. This work was supported by a grant from the National Institutes of Health (K22ES023815).

Date 2018-04-23

Depends R (>= 3.4.0)

License GPL-2

Encoding UTF-8

LazyData TRUE

Imports dplyr (>= 0.7.3),
tidyr (>= 0.7.1),
purrr (>= 0.2.3),
stringr (>= 1.2.0),
lubridate (>= 1.6.0),
sp (>= 1.2.5),
broom (>= 0.4.2),
ggplot2 (>= 2.2),
ggmap (>= 2.6.1),
maps (>= 3.2.0),
maptools (>= 0.9.2),
geosphere (>= 1.5.5),
rlang (>= 0.2.0),
rgeos (>= 0.3-23),
magrittr (>= 1.5),
utils (>= 3.4.1),
grDevices,
methods,
zoo (>= 1.8.0),
stats,
raster (>= 2.5.8),
readr (>= 1.1.1),
scales (>= 0.5.0),
tibble (>= 1.3.4),
colormap (>= 0.1.4),
rgdal (>= 1.2.8)

RoxygenNote 6.0.1

Suggests testthat,
knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

calculate_exposure . . . 2
california_shp . . . 5
chemical_list . . . 6
county_codes . . . 6
df_plot . . . 7
find_chemical_codes . . . 8
find_counties . . . 9
find_location_county . . . 9
find_product_name . . . 10
plot_application_timeseries . . . 11
plot_county_application . . . 12
plot_county_locations . . . 14
plot_exposure . . . 15
plot_locations_exposure . . . 17
pull_clean_pur . . . 18
pull_product_table . . . 22
pull_raw_pur . . . 23
pull_spdf . . . 24
purexposure . . . 25
spdf_to_df . . . 26
tibble_to_vector . . . 27
write_exposure . . . 27

Index **31**

`calculate_exposure` *Calculate exposure to active ingredients present in applied pesticides.*

Description

For a particular location, buffer radius, date range, and active ingredient or class of active ingredients, `calculate_exposure` calculates an estimate of exposure in kg of active ingredient per m^2.

Usage

```
calculate_exposure(clean_pur_df, location, radius, time_period = NULL,
  start_date = NULL, end_date = NULL, chemicals = "all",
  aerial_ground = FALSE, verbose = TRUE, ...)
```

Arguments

clean_pur_df	A data frame returned by pull_clean_pur that includes data for the county of your location (before running pull_clean_pur, you can use the find_location_county function to figure this out), the time period, and the active ingredients or chemical classes for which you want to calculate exposure.
location	A length-one character string. Either a California address including street name, city, state, and 5-digit zip code, or a pair of coordinates in the form "longitude, latitude".
radius	A numeric value greater than zero that gives the radius in meters defining the buffer around your location in which you would like to calculate exposure. For reference, the length and width of a PLS section is about 1,609 meters (1 mile). That of a township could range from about 9,656 to 11,265 meters (6-7 miles).
time_period	Optional. A character string giving a time period over which you would like to calculate exposure in days, weeks, months, or years. For example, if you enter "6 months" for time_period, calculate_exposure will calculate exposure for every six month period starting January 1 of the earliest year present in the clean_pur_df data frame. Start and end dates can be optionally specified with the start_date and end_date arguments. Alternatively, to calculate exposure over only one specified time period, you can leave this argument NULL and specify start and end dates.
start_date	Optional. "yyyy-mm-dd" specifying the start date for exposure estimation. This date should be present in the clean_pur_df data frame.
end_date	Optional. "yyyy-mm-dd" specifying the end date for exposure estimation. This date should be present in the clean_pur_df data frame.
chemicals	Either "all" or "chemical_class". The default is "all", which will calculate exposure to the summed active ingredients present in the clean_pur_df data frame. Enter "chemical_class" to calculate exposure to each of the chemical classes present in the chemical_class column of your clean_pur_df data frame.
aerial_ground	TRUE / FALSE for whether you would like to incorporate aerial/ground application into exposure calculations. If aerial_ground = TRUE, there should be an aerial_ground column in the input clean_pur_df data frame. There will be a value of exposure calculated for each chemical ("all" or by chemical class) and for each method of application: aerial or ground. The default is FALSE.
verbose	TRUE / FALSE for whether you would like a message to print out while the function is running. The default is TRUE.
...	Used internally.

Value

A list with five elements:

exposure A data frame with 9 columns: exposure, the estimate of exposure in kg/m^2, chemicals, (either "all", indicating that all active ingredients present in the clean_pur_df were summed or the chemical class(es) specified in the clean_pur_df data frame), start_date, end_date, aerial_ground, which can take values of "A" = aerial, "G" = ground, and "O" = others, (if the aerial_ground argument is FALSE, aerial_ground will be NA in the exposure data frame), location, radius, the radius in meters for the buffer extending from the location, and the longitude and latitude of the location.

meta_data A data frame with 12 columns and at least one row for every section or township intersected by the specified buffer extending from the given location. Columns include `pls`, giving either the Public Land Survey (PLS) section (9 characters long) or township (7 characters long), `chemicals`, `percent`, the percent that the PLS unit is overlapped by the buffer, `kg`, the total amount of kg applied for the specified chemicals and date range in that section or township, `kg_intersection`, the amount of kilograms applied multiplied by the percent of overlap, `start_date` and `end_date`, `aerial_ground`, which can take values of "A" (aerial), "G" (ground), or "O" (other), and will be `NA` if exposure calculations did not take aerial/ground application into account, `none_recorded`, logical for whether any pesticide application was recorded for the specified section or township, date range, and chemicals, `location`, and `radius`.

buffer_plot_df A data frame with 24 columns. Contains spatial plotting data for the buffer and overlapping sections or townships. You can use the `df_plot` function to quickly plot and get a rough idea of the area for which exposure was calculated, before moving on to other plot_* functions.

county_plot A ggplot2 plot showing the location of your specified buffer in the context of the county. Depending on if your `clean_pur_df` data frame was summed by section or township, the county will be shown with the relevant PLS units.

clean_pur_df The data frame supplied to the `clean_pur_df` argument, filtered to the county and date range for which exposure was calculated.

Note

- If the `time_period`, `start_date`, and `end_date` arguments are all left as NULL (their defaults), then exposure will be estimated across the entire date range of the `clean_pur_df` data frame.
- If you pulled PUR data from `pull_clean_pur` specifying `sum_application = TRUE` and `unit = "township"`, then exposure will be calculated based on townships. Using the `df_plot` function to plot the returned `buffer_plot` list element could be helpful to see the difference between calculating exposure based on sections or townships for a certain buffer radius.
- This function takes advantage of the Google Maps Geocoding API, and is limited by the standard usage limit of 2,500 free requests per day and 50 requests per second. https://developers.google.com/maps/documentation/geocoding/usage-limits

Examples

```
library(magrittr)

clean_pur <- readRDS(system.file("extdata", "fresno_clean.rds",
                                 package = "purexposure"))
fresno_spdf <- readRDS(system.file("extdata", "fresno_spdf.rds",
                                   package = "purexposure"))
exposure_list <- calculate_exposure(clean_pur, location = "-120.098794, 36.532866",
                                    radius = 3000, spdf = fresno_spdf)

# specify time intervals
exp_list2 <- calculate_exposure(clean_pur,
                                location = "13883 Lassen Ave, Helm, CA 93627",
                                radius = 3000,
                                time_period = "4 months")
exp_list2$exposure

# calculate exposure by township
```

```
clean_pur2 <- pull_clean_pur(1995, counties = "san bernardino",
                             sum_application = TRUE, unit = "township")
exp_list3 <- calculate_exposure(clean_pur2,
                                location = "-116.45, 34.96",
                                radius = 5000)
df_plot(exp_list3$buffer_plot)
exp_list3$county_plot

# calculate exposure by specified chemical classes
# this is an example of `none_recorded = TRUE`
chemical_class_df <- rbind(find_chemical_codes(2000, "methylene"),
                           find_chemical_codes(2000, "aldehyde")) %>%
   dplyr::rename(chemical_class = chemical)
exp_list4 <- pull_clean_pur(1995, "fresno",
                            chemicals = chemical_class_df$chemname,
                            sum_application = TRUE,
                            sum = "chemical_class",
                            chemical_class = chemical_class_df) %>%
   calculate_exposure(location = "13883 Lassen Ave, Helm, CA 93627",
                      radius = 1500,
                      chemicals = "chemical_class")
exp_list4$meta_data

# incorporate aerial/ground application information
exp_list5 <- pull_clean_pur(2000, "yolo") %>%
   calculate_exposure(location = "-121.9018, 38.7646",
   radius = 2500,
   aerial_ground = TRUE)
exp_list5$exposure
```

`california_shp` *California state SpatialPolygonsDataFrame object.*

Description

A SpatialPolygonsDataFrame object for the outline of the state of California. Downloaded and subset from the 2016 U.S. Census Cartographic Boundary Shapefile for states.

Usage

```
california_shp
```

Format

A SpatialPolygonsDataFrame object at the 1:20,000,000 resolution level.

Source

https://www.census.gov/geo/maps-data/data/cbf/cbf_state.html

chemical_list *California Pesticide Use Report chemical codes.*

Description

A list of data frames (one for each year from 1990 through 2015) containing California Department of Pesticide Regulation chemical codes and names used to identify active ingredients in Pesticide Use Report data. "chemical.txt" files for each year were pulled from the .zip files "pur1990.zip" through "pur2015.zip" found here: ftp://transfer.cdpr.ca.gov/pub/outgoing/pur_archives

Usage

```
chemical_list
```

Format

A list of 26 elements. Each element is a data frame with a variable number of rows ranging from 3,579 to 3,934 and two columns:

chem_code An integer giving the chemical code. This uniquely identifies a chemical within a year.

chemname A character vector giving common chemical name for each active ingredient. These are usually listed on the pesticide product label.

Source

ftp://transfer.cdpr.ca.gov/pub/outgoing/pur_archives

county_codes *California Pesticide Use Report county codes.*

Description

A data frame containing California county names and corresponding PUR and FIPS codes. PUR county codes are unique to PUR datasets, and FIPS county codes are U.S. Federal Information Processing Standard codes. The file with county names and PUR codes, "county.txt", was pulled from the .zip file "pur2000.zip" found here: ftp://transfer.cdpr.ca.gov/pub/outgoing/pur_archives. FIPS codes were pulled from here: http://www2.census.gov/geo/docs/reference/cenpop2010/county/CenPop2010_Mean_CO.txt

Usage

```
county_codes
```

Format

A data frame with 58 rows and three columns:

county_name A character vector giving California county names.

pur_code A character vector giving two-digit PUR county codes (ranging from "01" through "58") corresponding to each California county. Note: these codes are unique to California PUR datasets; they do not correspond to FIPS codes.

fips_code A character vector giving six-digit FIPS county codes corresponding to each California county.

Source

ftp://transfer.cdpr.ca.gov/pub/outgoing/pur_archives http://www2.census.gov/geo/docs/reference/cenpop2010/county/CenPop2010_Mean_CO.txt

---

df_plot *Plot data frame spatial objects.*

Description

`df_plot` plots a data frame spatial object. (A SpatialPolygonsDataFrame that has been "tidied" using the broom package.) Meant to be analogous to the ease of using plot() to quickly view a SpatialPolygonDataFrame object.

Usage

```
df_plot(df)
```

Arguments

`df`	A data frame returned from the `spdf_to_df` function.

Value

A ggplot2 plot of the county.

Examples

```
library(magrittr)

fresno <- readRDS(system.file("extdata", "fresno_spdf.rds", package = "purexposure"))
fresno %>% spdf_to_df %>% df_plot()

pull_spdf("fresno") %>% spdf_to_df() %>% df_plot()
pull_spdf("san diego", "township") %>%
   spdf_to_df() %>%
   df_plot()
```

`find_chemical_codes` *Pull active ingredient chemical codes from PUR Chemical Lookup Tables.*

Description

For a vector of chemical names, `find_chemical_codes` returns a data frame with corresponding chemical codes from the PUR Chemical Lookup Table for a given year. This function uses pattern matching to return results. As a starting place, or for more thorough classifications, see the CA Department of Pesticide Regulation's Summary of Pesticide Use Report Data, Indexed by Chemical (2008): http://www.cdpr.ca.gov/docs/pur/pur08rep/chmrpt08.pdf

Usage

```
find_chemical_codes(years, chemicals = "all", by_year = FALSE)
```

Arguments

`years`	A vector of four-digit numeric years in the range of 1990 to 2015. Indicates the years in which you would like to match chemical codes.
`chemicals`	A string or vector of strings giving search terms of chemicals to match with active ingredients present in pesticides applied in the given year. The default value is "all", which returns codes for all active ingredients applied in a given year.
`by_year`	TRUE / FALSE for whether you would like PUR Chemical Lookup Tables separated by year (in a 'year' column). If 'by_year' is 'FALSE', the default, a data frame is returned with unique results from all years given in the 'years' argument.

Value

A data frame:

chem_code An integer value with chemical codes corresponding to each active ingredient. `chem_code` values are used to later filter raw PUR data sets.

chemname A character string giving unique active ingredients corresponding to each search term.

chemical A character string with search terms given in the `chemicals` argument. Not included if the 'chemicals' argument is set to its default value of "all".

year Included if 'by_year' is set to 'TRUE'.

Note

The PUR Chemical Lookup Table for a year lists all active ingredients present in applied pesticides across the state of California. Therefore, PUR data for a particular county may not include records for active ingredients returned by `find_chemical_codes` for the same year.

Examples

```
find_chemical_codes(2000, "methyl bromide")
find_chemical_codes(1995, c("ammonia", "benzene"))
```

find_counties *Find California county codes or names.*

Description

Given a vector of counties, find_counties returns either PUR county codes or names.

Usage

```
find_counties(counties, return = "pur_codes")
```

Arguments

counties	A vector of character strings giving either a county names, two digit PUR county codes, or six-digit FIPS county codes. Not case sensitive. California names and county codes as they appear in PUR data sets can be found in the county_codes data set available with this package.
return	Either "pur_codes" to return PUR county codes (the default), "fips_codes" to return FIPS county codes, or "names" to return county names.

Value

If return = "pur_codes", a vector of two-character strings giving the corresponding PUR county codes. If return = "fips_codes", a vector of six-digit character strings giving the corresponding FIPS county codes. If return = "names", a vector of county names.

Examples

```
find_counties(c("01", "06005", "el dorado"))
find_counties(c("01", "06005", "el dorado"), return = "fips_codes")
find_counties(c("01", "06005", "el dorado"), return = "names")
```

find_location_county *Find the county from an address or coordinate pair.*

Description

Given a California address or longitude/latitude coordinates, find_location_county returns the corresponding California county or PUR code.

Usage

```
find_location_county(locations, return = "name", ...)
```

Arguments

locations	A vector of character strings. Each location should be either a California address including street name, city, state, and 5-digit zip code, or a pair of coordinates in the form "longitude, latitude".
return	Either "name" to return county name (the default), "pur_code" to return PUR county code, or "fips_code" to return the FIPS county code.
...	Used internally.

Value

A character string giving the California county where the address or coordinate pair given in `location` is located.

Examples

```
address <- "13883 Lassen Ave, Helm, CA 93627"
long_lat <- c("-120.09789, 36.53379")
find_location_county(c(address, long_lat))
```

`find_product_name` *Find pesticide product names and registration numbers from PUR Product Lookup Tables.*

Description

For a vector of years and product search terms, `find_product_name` returns a data frame with corresponding product registration numbers, `prodno`, indicator codes, and product names.

Usage

```
find_product_name(years, products = "all", quiet = FALSE, by_year = FALSE,
  ...)
```

Arguments

`years`	A vector of four digit years in the range of 1990 to 2015.
`products`	A character string or a vector of character strings with pesticide product names that you would like to search for. Not case sensitive. The default is "all", which will return all pesticide products applied for a given year.
`quiet`	TRUE / FALSE indicating whether you would like a message and progress bar printed for the product table that is downloaded. The default value is FALSE.
`by_year`	TRUE / FALSE for whether you would like PUR Product Lookup Tables separated by year (in a 'year' column). If 'by_year' is 'FALSE', the default, a data frame is returned with unique results from all years given in the 'years' argument.
`...`	Used internally.

Details

Product tables are pulled by year from the CDPR's FTP server. Downloaded tables are saved in a temporary environment, which is deleted at the end of the current R session.

Value

A data frame with seven columns:

prodno The CA registration number. Can be matched with the prodno in a raw or cleaned PUR data set.

prodstat_ind Character. An indication of product registration status:

- A = Active
- B = Inactive
- C = Inactive, Not Renewed
- D = Inactive, Voluntary Cancellation
- E = Inactive, Cancellation
- F = Inactive, Suspended
- G = Inactive, Invalid Data
- H = Active, Suspended

product_name Character. The name of the product taken from the registered product label. May have been modified by DPR's Registration Branch to ensure uniqueness.

signlwrd_ind Integer. The signal word printed on the front of the product label:

- 1 = Danger (Poison)
- 2 = Danger (Only)
- 3 = Warning
- 4 = Caution
- 5 = None

product Product name search terms.

year The year for which product table information was pulled. Included if 'by_year' is set to TRUE.

Examples

```
prod_df <- find_product_name(2000, "mosquito")
prod_df2 <- find_product_name(2010, c("insecticide", "rodenticide"))
```

---

plot_application_timeseries

Plot time series of active ingredients in applied pesticides.

Description

plot_application_timeseries returns a ggplot2 time series plot of pesticides present in a pull_clean_pur data frame. You can choose whether to facet the time series by active ingredient (chemname) or by chemical_class.

Usage

```
plot_application_timeseries(clean_pur_df, facet = FALSE, axes = "fixed")
```

Arguments

clean_pur_df	A data frame returned from pull_clean_pur.
facet	TRUE / FALSE for whether you would like time series plots to be faceted by unique chemname or chemical_class column values. If facet = FALSE (the default), all active ingredients present in the data set will be summed per day.
axes	A character string passed on to the scales argument of ggplot2::facet_wrap ("fixed", "free", "free_x", or "free_y"). The default is "fixed".

Value

A ggplot2 object.

Examples

```
library(magrittr)
readRDS(system.file("extdata", "fresno_clean.rds", package = "purexposure")) %>%
    plot_application_timeseries()

pull_clean_pur(1990:1992, "fresno") %>%
    dplyr::filter(chemname %in% toupper(c("methyl bromide", "sulfur"))) %>%
    plot_application_timeseries(facet = TRUE)
```

---

plot_county_application

Plot pesticide application by county.

Description

plot_county_application returns a plot of applied pesticides (either the sum of all active ingredients present in the input pull_clean_pur data frame, a specified chemical class, or a specified active ingredient). Application is summed by section or township. PLS units can be shaded by amount or by percentile.

Usage

```
plot_county_application(clean_pur_df, county = NULL, pls = NULL,
  color_by = "amount", percentile = c(0.25, 0.5, 0.75), start_date = NULL,
  end_date = NULL, chemicals = "all", fill = "viridis", crop = FALSE,
  alpha = 1, ggmap_background = TRUE, ...)
```

Arguments

clean_pur_df	A data frame returned by pull_clean_pur.
county	Optional. If your clean_pur_df data frame contains data for multiple counties, this argument specifies which county you would like to plot application for. Either a PUR county name or county code. California names and county codes as they appear in PUR data sets can be found in the county_codes data set available with this package.

pls	Optional. Either "section" or "township". If your clean_pur_df data frame has both a section and township column, the pls argument specifies which pls unit you would like to plot application for (the default in this case is "section"). If you pulled data specifying unit = "township", application will be plotted by township.
color_by	Either "amount" (the default) or "percentile". Specifies whether you would like application amounts to be colored according to amount, resulting in a gradient legend, or by the percentile that they fall into for the given data set and date range. You can specify percentile cutpoints with the percentile argument.
percentile	A numeric vector in (0, 1) specifying percentile cutpoints if color_by = "percentile". The default is c(0.25, 0.5, 0.75), which results in four categories: < 25th percentile, >= 25th to < 50th, >= 50th to < 75th, and >= 75th.
start_date	Optional. "yyyy-mm-dd" giving a starting date for the date range that you would like to map application for. The default is to plot application for the entire date range in your clean_pur_df data frame.
end_date	Optional. "yyyy-mm-dd" giving an ending date for the date range that you would like to plot application for. The default is to plot application for the entire date range in your clean_pur_df data frame.
chemicals	Either "all" (the default) to plot summed active ingredients present in your clean_pur_df data frame, a chemical class present in the chemical_class column of the clean_pur_df data frame, or a specific active ingredient present in the chemname column of the clean_pur_df data frame.
fill	A palette from the colormap package. The default is "viridis". To see colormap palette options, visit https://bhaskarvk.github.io/colormap/ or run colormap::colormaps.
crop	TRUE / FALSE for whether you would like your plot zoomed in on sections or townships with recorded application data.
alpha	A number in [0,1] specifying the transparency of fill colors. Numbers closer to 0 will result in more transparency. The default is 1.
ggmap_background	TRUE / FALSE for whether you would like a ggmap background.
...	Used internally.

Value

A list with three elements:

map A plot of the county with application summed by section or township and colored by amount or by percentile.

data A data frame with the plotted application data.

cutoff_values A data frame with two columns: percentile and kg, giving the cut points for each percentile in the clean_pur_df for the specified chemicals. This element of the list is not returned if color_by = "amount".

Examples

```
library(magrittr)

fresno_spdf <- readRDS(system.file("extdata", "fresno_spdf.rds",
                                   package = "purexposure"))
fresno_clean <- readRDS(system.file("extdata", "fresno_clean.rds",
```

```
                                    package = "purexposure"))
fresno_list <- fresno_clean %>% plot_county_application(spdf = fresno_spdf)

# plot all active ingredients
fresno_df <- pull_clean_pur(2000:2001, "fresno")
fresno_list <- plot_county_application(fresno_df,
                                       color_by = "percentile",
                                       percentile = c(0.2, 0.4, 0.6, 0.8))
fresno_list$map
head(fresno_list$data)
fresno_list$cutoff_values

# plot a specific active ingredient
fresno_list2 <- plot_county_application(fresno_df, pls = "township",
                                        chemicals = "sulfur",
                                        fill = "plasma")
fresno_list2$map

# plot a chemical class
chemical_class_df <- purrr::map2_dfr(2010, c("methidathion", "parathion",
                                             "naled", "malathion",
                                             "trichlorfon"),
                                     find_chemical_codes) %>%
     dplyr::mutate(chemical_class = "organophosphates") %>%
     dplyr::select(-chemical)
op_yuba <- pull_clean_pur(2010, "yuba",
                          chemicals = chemical_class_df$chemname,
                          verbose = F, sum_application = T,
                          sum = "chemical_class",
                          chemical_class = chemical_class_df) %>%
   plot_county_application()
op_yuba$map
```

`plot_county_locations` *Plot a county's location in California.*

Description

`plot_county_locations` returns one or multiple plots with county locations in California given either a vector of county names or codes, or a PUR data frame with a `county_cd`, `county_name`, `pur_code`, or `fips_code` column (A data frame returned from either `pull_pur_file`, `pull_raw_pur`, or `pull_clean_pur`).

Usage

```
plot_county_locations(counties_or_df, separate_plots = FALSE,
  fill_color = "red", alpha = 0.5, ...)
```

Arguments

`counties_or_df`	A character vector of county names, pur codes, or fips codes. You can use the `county_codes` data set included with this package to check out PUR county

	names and codes. This argument can also be a data frame with a county_cd, county_name, pur_code, or fips_code column. If you provide a data frame, a plot for every county with data in that data set will be output.
separate_plots	TRUE / FALSE. If you provided multiple counties, whether you would like county outlines plotted in the same plot (FALSE), or if you would like separate plots returned in a list (TRUE). The default is FALSE.
fill_color	A character string giving either a ggplot2 color or a hex color code ("#0000FF", for example). The default is "red".
alpha	A number in [0,1] specifying the transparency of the fill color. Numbers closer to 0 will result in more transparency. The default is 0.5.
...	Used internally.

Value

A ggplot or a list of ggplots of Califnornia with shaded-in counties. List element names correspond to county names.

Examples

```
fresno_spdf <- readRDS(system.file("extdata", "fresno_spdf.rds",
                                   package = "purexposure"))
plot_county_locations("fresno", spdf = fresno_spdf)

plot_county_locations("fresno")

pur_df <- pull_clean_pur(1990, counties = c("01", "05", "12"))
plot_county_locations(pur_df)

plot_list <- plot_county_locations(c("san bernardino", "ventura"),
                                   separate_plots = TRUE)
names(plot_list)
plot_list[[1]]
plot_list[[2]]
```

---

plot_exposure *Plot exposure to applied pesticides at a location.*

Description

plot_exposure returns a plot of pesticide application in the PLS units intersected by a buffer for each combination of time period, applied active ingredients, and application method relevant for the exposure values returned from calculate_exposure.

Usage

```
plot_exposure(exposure_list, color_by = "amount",
  buffer_or_county = "county", percentile = c(0.25, 0.5, 0.75),
  fill = "viridis", alpha = 0.7, pls_labels = FALSE,
  pls_labels_size = 4)
```

Arguments

exposure_list	A list returned from calculate_exposure.
color_by	Either "amount" (the default) or "percentile". Specifies whether you would like application amounts to be colored according to amount, resulting in a gradient legend, or by the percentile that they fall into for the given data set and date range. You can specify percentile cutpoints with the percentile argument.
buffer_or_county	Either "county" (the default) or "buffer". Specifies whether you would like colors to be scaled according to the limits of application within the buffer, or in the county for the same time period, chemicals, and method of application.
percentile	A numeric vector in (0, 1) specifying percentile cutpoints if color_by = "percentile". The default is c(0.25, 0.5, 0.75), which results in four categories: < 25th percentile, >= 25th to < 50th, >= 50th to < 75th, and >= 75th.
fill	A palette from the colormap package. The default is "viridis". To see colormap palette options, visit https://bhaskarvk.github.io/colormap/ or run colormap::colormaps.
alpha	A number in [0,1] specifying the transparency of fill colors. Numbers closer to 0 will result in more transparency. The default is 0.7.
pls_labels	TRUE / FALSE for whether you would like sections or townships to be labeled with their PLS ID. The default is FALSE.
pls_labels_size	A number specifying the size of PLS labels. The default is 4.

Value

A list with the following elements:

maps A list of plots. One plot for each exposure value returned in the exposure element of the calculate_exposure list.

pls_data A list of data frames with 12 columns: pls, giving the PLS ID, percent, the buffer, kg, the amount of kg of pesticides applied in that PLS unit for the relevant time period, chemicals, and application method, kg_intersection, kg multiplied by percent (this is the value that is plotted), start_date, end_date, chemicals, aerial_ground, which give the time period, chemicals, and application method for each plot/exposure estimate, none_recorded, location, radius (m), and area (m^2).

cutoff_values A list of data frames with two columns: percentile and kg giving the cutoff values for each percentile. Only returned if color_by = "percentile".

Examples

```
library(magrittr)

fresno_list <- readRDS(system.file("extdata", "exposure_ex.rds",
                                   package = "purexposure")) %>% plot_exposure()

tulare_list <- pull_clean_pur(2010, "tulare") %>%
   calculate_exposure(location = "-119.3473, 36.2077", radius = 3500) %>%
   plot_exposure()
names(tulare_list)
tulare_list$maps
tulare_list$pls_data
tulare_list$exposure
```

```
# return one plot, pls_data data frame, exposure row, and cutoff_values
# data frame for each exposure combination

dalton_list <- pull_clean_pur(2000, "modoc") %>%
    calculate_exposure(location = "-121.4182, 41.9370",
                       radius = 4000,
                       time_period = "6 months",
                       aerial_ground = TRUE) %>%
    plot_exposure(fill = "plasma")
do.call("rbind", dalton_list$exposure)
# one map for each exposure value (unique combination of chemicals,
# dates, and aerial/ground application)
dalton_list$maps[[1]]
dalton_list$maps[[2]]
dalton_list$maps[[3]]
dalton_list$maps[[4]]
dalton_list$maps[[5]]
dalton_list$maps[[6]]

# exposure to a particular active ingredient
# plot percentile categories instead of amounts
chemical_df <- rbind(find_chemical_codes(2009, c("metam-sodium"))) %>%
     dplyr::rename(chemical_class = chemical)

santa_maria <- pull_clean_pur(2008:2010, "santa barbara",
                              chemicals = chemical_df$chemname,
                              sum_application = TRUE,
                              sum = "chemical_class",
                              chemical_class = chemical_df) %>%
     calculate_exposure(location = "-119.6122, 34.90635",
                        radius = 3000,
                        time_period = "1 year",
                        chemicals = "chemical_class") %>%
     plot_exposure(color_by = "percentile")
do.call("rbind", santa_maria$exposure)
santa_maria$maps[[1]]
santa_maria$maps[[2]]
santa_maria$maps[[3]]

# scale colors based on buffer or county
clotho <- pull_clean_pur(1996, "fresno") %>%
  dplyr::filter(chemname == "SULFUR") %>%
  calculate_exposure(location = "-119.6082, 36.7212",
                     radius = 1500)

plot_exposure(clotho, "amount", buffer_or_county = "county", pls_labels = TRUE)$maps
plot_exposure(clotho, "amount", buffer_or_county = "buffer", pls_labels = TRUE)$maps
```

---

plot_locations_exposure

Plot exposure for multiple locations in a county.

Description

plot_locations_exposure returns a plot of exposure to applied pesticides for multiple locations in a given county.

Usage

```
plot_locations_exposure(exposure_df, section_township = "section",
  fill = "viridis", alpha = 1, ...)
```

Arguments

exposure_df	A data frame returned from write_exposure with 10 columns, including exposure, location, radius, longitude, and latitude. This data frame should be filtered so that there is one exposure value per location. (This could also be the 'exposure' element returned from 'calculate_exposure'.)
section_township	Either "section" (the default) or "township". Specifies which PLS unit to plot the county by.
fill	A palette from the colormap package. The default is "viridis". To see colormap palette options, visit https://bhaskarvk.github.io/colormap/ or run colormap::colormaps.
alpha	A number in [0,1] specifying the transparency of fill colors. Numbers closer to 0 will result in more transparency. The default is 1.
...	Used internally.

Value

A plot with one point per location, colored by each location's corresponding exposure value.

Examples

```
fresno <- purexposure::fresno_clean
df <- data.frame(location = c("295 West Saginaw Ave., Caruthers, CA 93609",
                              "55190 Point Rd., Big Creek, CA 93605"),
                 start_date = "2000-01-01", end_date = "2000-12-31")
temp_dir <- tempdir()
write_exposure(fresno, df, 3000, temp_dir)
exp_df <- readRDS(paste0(temp_dir, "/exposure_df.rds"))
plot_locations_exposure(exp_df)
```

---

pull_clean_pur *Pull cleaned PUR data by counties, years, and active ingredients.*

Description

pull_clean_pur returns a data frame of cleaned Pesticide Use Report data filtered by counties, years, and active ingredients. Active ingredients or chemical classes present in applied pesticides can be summed by either Public Land Survey (PLS) section or township.

Usage

```
pull_clean_pur(years = "all", counties = "all", chemicals = "all",
  sum_application = FALSE, unit = "section", sum = "all",
  chemical_class = NULL, aerial_ground = TRUE, verbose = TRUE,
  quiet = FALSE, ...)
```

Arguments

years	A four-digit numeric year or vector of years in the range of 1990 to 2015. Indicates the years for which you would like to pull PUR data sets. `years == "all"` will pull data from 1990 through 2015.
counties	A vector of character strings giving either a county name, two digit PUR county codes, or six-digit FIPS county codes for each county. Not case sensitive. California names, county codes as they appear in PUR data sets, and FIPS county codes can be found in the `county_codes` data set available with this package. For example, to return data for Alameda county, enter either "alameda", "01", or "06001" for the `counties` argument. `counties = "all"` will return data for all 58 California counties (this will take a while to run).
chemicals	A string or vector of strings giving search terms of chemicals to match with active ingredients present in pesticides applied in the given years. The default value is "all", which returns records for all active ingredients applied in a given year. See the CDPR's Summary of PUR Data document here: http://www.cdpr.ca.gov/docs/pur/pur08rep/chmrpt08.pdf for comprehensive classifications of active ingredients.
sum_application	TRUE / FALSE indicating if you would like to sum the amounts of applied active ingredients by day, the geographic unit given in `unit`, and by either active ingredients or chemical class (indicated by `sum` and `chemical_class`). The default value is FALSE.
unit	A character string giving either "section" or "township". Specifies whether applications of each active ingredient should be summed by California section (the default) or by township. Only used if `sum_application` is `TRUE`.
sum	A character string giving either "all" (the default) or "chemical_class". If `sum_application = TRUE`, `sum` indicates whether you would like to sum across all active ingredients, giving an estimation of the total pesticides applied in a given section or township ("all"), or by a chemical class specified in a data frame given in the argument `chemical_class`.
chemical_class	A data frame with only three columns: `chem_code`, `chemname`, and `chemical_class`. `chem_code` should have integer values giving PUR chemical codes, and `chemname` should have character strings with corresponding PUR chemical names (these can be searched for using the `find_chemical_codes` function or with the `chemical_list` data set included with this package). The `chemical_class` column should have character strings indicating the chemical class corresponding to each `chem_code`. The `chemical_class` for a group of active ingredients should be decided upon by the user. Only used if `sum = "chemical_class"`. See the CDPR's Summary of PUR Data document here: http://www.cdpr.ca.gov/docs/pur/pur08rep/chmrpt08.pdf for comprehensive classifications of active ingredients.
aerial_ground	TRUE / FALSE indicating if you would like to retain aerial/ground application data ("A" = aerial, "G" = ground, and "O" = other.) The default is TRUE.

verbose	TRUE / FALSE indicating whether you would like a single message printed indicating which counties and years you are pulling data for. The default value is TRUE.
quiet	TRUE / FALSE indicating whether you would like a message and progress bar printed for each year of PUR data that is downloaded. The default value is FALSE.
...	Used internally.

Details

PUR data sets are pulled by county from the CDPR's FTP server. Downloaded PUR data sets are saved in a temporary environment, which is deleted at the end of the current R session.

Value

A data frame:

chem_code An integer value giving the PUR chemical code for the active ingredient applied. Not included if `sum_application = TRUE` and `sum = "chemical_class"`.

chemname A character string giving PUR chemical active ingredient names. Unique values of `chemname` are matched with terms provided in the `chemicals` argument. Not included if `sum_application = TRUE` and `sum = "chemical_class"`.

chemical_class If `sum_application = TRUE` and `sum = "chemical_class"`, this column will give values of the `chemical_class` column in the input `chemical_class` data frame. If there are active ingredients pulled based on the `chemicals` argument that are not present in the `chemical_class` data frame, these chemicals will be summed under the class "other".

kg_chm_used A numeric value giving the amount of the active ingredient applied (kilograms).

section A string nine characters long indicating the section of application. PLS sections are uniquely identified by a combination of base line meridian (S, M, or H), township (01-48), township direction (N or S), range (01-47), range direction (E or W) and section number (01-36). This column is not included if `sum_application = TRUE` and `unit = "township"`.

township A string seven characters long indicating the township of application. PLS townships are uniquely identified by a combination of base line meridian (S, M, or H), township (01-48), township direction (N or S), range (01-47), and range direction (E or W).

county_name A character string giving the county name where application took place.

pur_code A string two characters long giving the PUR county code where application took place.

fips_code A string six characters long giving the FIPS county code where application took place.

date The date of application (yyyy-mm-dd).

aerial_ground A character giving the application method. "A" = aerial, "G" = ground, and "O" = other. Not included if `aerial_ground = FALSE`.

use_no A character string identifying unique application of an active ingredient across years. This value is a combination of the raw PUR `use_no` column and the year of application. Not included if `sum_appliction = TRUE`.

outlier If the amount listed in `kg_chm_used` has been corrected for large amounts entered in error, this column lists the raw value of recorded kilograms of applied chemicals. Otherwise `NA`. The algorithm for identifying and replacing outliers was developed based on methods used by Gunier et al. (2001). Please see the package vignette for more detail regarding these methods. Not included if `sum_application = TRUE`.

prodno Integer. The California Registration Number for the applied pesticide (will be repeated for different active ingredients present in the product). You can match product registration numbers with product names, which can be pulled using the `pull_product_table` function. This column is not returned if `sum_application = TRUE`.

Note

- The `chemical_list` data frame for a particular year lists active ingredients present in applied pesticides across the state of California. Therefore, PUR data for a particular county may not include records for active ingredients listed in the `chemical_list` data set for the same year.
- To pull raw PUR data, see the `pull_raw_pur` function. For documentation of raw PUR data, see the Pesticide Use Report Data User Guide & Documentation document published by the California Department of Pesticide Regulation. This file is saved as "cd_doc.pdf" in any "pur[year].zip" file between 1990 and 2015 found here: ftp://transfer.cdpr.ca.gov/pub/outgoing/pur_archives/.
- If this function returns an error (because the FTP site is down, for example), check your working directory. You may need to change it back from a temporary directory.

Examples

```
library(magrittr)


df <- pull_clean_pur(years = 2000:2001,
                     counties = c("06001", "29", "riverside"),
                     chemicals = "methylene",
                     aerial_ground = TRUE)

# filter to active ingredients present in particular products
prod_nos <- find_product_name(2003, "insecticide") %>%
    dplyr::select(prodno) %>%
    tibble_to_vector()

df2 <- pull_clean_pur(2003, "10") %>%
    dplyr::filter(prodno %in% prod_nos)

# Sum application by active ingredients
df3 <- pull_clean_pur(years = 2009:2010,
                      counties = c("01", "29", "riverside"),
                      unit = "township",
                      sum_application = TRUE)

# Or by chemical classes
chemical_class_df <- rbind(find_chemical_codes(2000, "methylene"),
                           find_chemical_codes(2000, "aldehyde")) %>%
   dplyr::rename(chemical_class = chemical)

df4 <- pull_clean_pur(years = 1995,
                      counties = "fresno",
                      chemicals = chemical_class_df$chemname,
                      sum_application = TRUE,
                      sum = "chemical_class",
                      unit = "township",
                      chemical_class = chemical_class_df)
```

`pull_product_table` *Pull PUR Product Table.*

Description

This function pulls a California Department of Pesticide Regulation Product Table for a vector of years.

Usage

```
pull_product_table(years, quiet = FALSE)
```

Arguments

`years`	A vector of four digit years in the range of 1990 to 2015.
`quiet`	TRUE / FALSE indicating whether you would like a message and progress bar printed for the product table that is downloaded. The default value is FALSE.

Details

Product tables are pulled by year from the CDPR's FTP server. Downloaded tables are saved in a temporary environment, which is deleted at the end of the current R session.

Value

A data frame with four columns:

prodno Integer. The California Registration number for the pesticide product. This corresponds to the `prodno` column in a raw or cleaned PUR data set returned from `pull_raw_pur` or `pull_clean_pur`.

prodstat_ind Character. An indication of product registration status:

- A = Active
- B = Inactive
- C = Inactive, Not Renewed
- D = Inactive, Voluntary Cancellation
- E = Inactive, Cancellation
- F = Inactive, Suspended
- G = Inactive, Invalid Data
- H = Active, Suspended

product_name Character. The name of the product taken from the registered product label. May have been modified by DPR's Registration Branch to ensure uniqueness.

signlwrd_ind Integer. The signal word printed on the front of the product label:

- 1 = Danger (Poison)
- 2 = Danger (Only)
- 3 = Warning
- 4 = Caution
- 5 = None

year Integer. Four digit year indicating the year for which data was pulled.

Examples

```
prod_95 <- pull_product_table(1995)
```

`pull_raw_pur` *Pull raw PUR data by counties and years.*

Description

`pull_raw_pur` pulls a raw PUR data set for a given year and vector of California counties.

Usage

```
pull_raw_pur(years = "all", counties = "all", verbose = TRUE,
  quiet = FALSE)
```

Arguments

`years`	A four-digit numeric year or vector of years in the range of 1990 to 2015. Indicates the years for which you would like to pull PUR data sets. `years == "all"` will pull data from 1990 through 2015.
`counties`	A vector of character strings giving either a county name, two digit PUR county codes, or six-digit FIPS county codes for each county. Not case sensitive. California names, county codes as they appear in PUR data sets, and FIPS county codes can be found in the `county_codes` data set available with this package. For example, to return data for Alameda county, enter either "alameda", "01", or "06001" for the `counties` argument. `counties = "all"` will return data for all 58 California counties (this will take a while to run).
`verbose`	TRUE / FALSE indicating whether you would like a single message printed indicating which counties and years you are pulling data for. The default value is TRUE.
`quiet`	TRUE / FALSE indicating whether you would like a message and progress bar printed for each year of PUR data that is downloaded. The default value is FALSE.

Details

PUR data sets are pulled by county from the CDPR's FTP server. Downloaded PUR data sets are saved in a temporary environment, which is deleted at the end of the current R session.

Value

A data frame with 33 columns. Different years and counties for which data was pulled are indicated by `applic_dt` and `county_cd`, respectively.

Note

- For documentation of raw PUR data, see the Pesticide Use Report Data User Guide & Documentation document published by the California Department of Pesticide Regulation. This file is saved as "cd_doc.pdf" in any "pur[year].zip" file between 1990 and 2015 found here: ftp://transfer.cdpr.ca.gov/pub/outgoing/pur_archives/.
- If this function returns an error (because the FTP site is down, for example), check your working directory. You may need to change it back from a temporary directory.

Examples

```
df <- pull_raw_pur(years = 2000, counties = "fresno")
df2 <- pull_raw_pur(years = c(2000, 2010), counties = c("butte", "15", "06001"))
df3 <- pull_raw_pur(years = 2015, counties = c("colusa"))
```

---

pull_spdf *Pull California county SpatialPolygonsDataFrame.*

Description

pull_spdf pulls either the section or township-level SpatialPolygonsDataFrame from a county's Geographic Information System (GIS) shapefile.

Usage

```
pull_spdf(county, section_township = "section", quiet = FALSE)
```

Arguments

county	A character string giving either a county name, two digit PUR county code, or six-digit FIPS county code. Not case sensitive. California names and county codes as they appear in PUR data sets can be found in the county_codes data set available with this package.
section_township	Either "section" (the default) or "township". Specifies whether you would like to pull a section- or township-level SpatialPolygonsDataFrame.
quiet	TRUE / FALSE indicating whether you would like a message and progress bar printed for the shapefile that is downloaded. The default value is FALSE.

Details

SpatialPolygonsDataFrame objects are pulled by county from the CDPR's FTP server. Downloaded SpatialPolygonsDataFrame objects are saved in a temporary environment, which is deleted at the end of the current R session.

Value

A SpatialPolygonsDataFrame object.

Source

SpatialPolygonDataFrame objects are downloaded from GIS shapefiles provided by the California Department of Pesticide Regulation: http://www.cdpr.ca.gov/docs/emon/grndwtr/gis_shapefiles.htm

Note

If this function returns an error (because the FTP site is down, for example), check your working directory. You may want to change it back from a temporary directory.

Examples

```
fresno_shp <- pull_spdf("fresno")
fresno_shp %>% spdf_to_df() %>% df_plot()
del_norte_shp <- pull_spdf("08", "township")
del_norte_shp %>% spdf_to_df() %>% df_plot()
```

`purexposure` *purexposure: A package for working with CA Pesticide Use Registry data.*

Description

The `purexposure` package provides functions to pull data from California's Pesticide Use Registry (PUR), as well as to calculate exposure to and visualize active ingredients present in applied pesticides. Functions are categorized into `find_*`, `pull_*`, `calculate_*`, `plot_*`, and `write_*`. These are the functions from each category:

find_* functions

`find_` functions help with searches of PUR chemical, county, and product codes.

- `find_chemical_codes`: Pull active ingredient chemical codes from PUR Chemical Lookup Tables.
- `find_counties`: Find California county codes or names.
- `find_location_county`: Find the counties of addresses or coordinates.
- `find_product_name`: Find Pesticide Product names and registration numbers from PUR Product Lookup Tables.

pull_* functions

`pull_` functions facilitate downloading data from the CA Department of Pesticide Regulation's website.

- `pull_raw_pur`: Pull raw PUR data by counties and years.
- `pull_clean_pur`: Pull cleaned PUR data by counties, years, and active ingredients.
- `pull_product_table`: Pull PUR Product Tables for a vector of years.
- `pull_spdf`: Pull section or township-level SpatialPolygonsDataFrame for a county.

calculate_* function

The calculate_exposure function calculates exposure (in kg/m^2) to applied pesticides for a given location, buffer extending from that location, time period, and active ingredients.

plot_* functions

plot_ functions help with visualizations of application.

- plot_county_application: Plot pesticide application by county, summed by section or township.
- plot_county_locations: Plot county locations in California.
- plot_exposure: Plot exposure to applied pesticides at a particular location.
- plot_application_timeseries: Plot time series of active ingredients in applied pesticides.
- plot_locations_exposure: Plot exposure for multiple locations in a given county.

write_* function

The write_exposure function calculates exposure for multiple locaitons and writes out files (exposure values, meta data, and plots) to a specified directory.

---

spdf_to_df *Convert county SpatialPolygonsDataFrame to a tidy data frame.*

Description

spdf_to_df converts a SpatialPolygonsDataFrame object returned from the pull_spdf function to a data frame.

Usage

```
spdf_to_df(spdf)
```

Arguments

spdf	A SpatialPolygonsDataFrame object returned from the pull_spdf function.

Value

A data frame with 24 columns if the spdf object is on the section level and 23 columns if the spdf object is on the township level.

Examples

```
library(magrittr)

df <- readRDS(system.file("extdata", "fresno_spdf.rds", package = "purexposure")) %>%
   spdf_to_df()

df <- pull_spdf("frenso") %>% spdf_to_df()
df2 <- pull_spdf("sonoma") %>% spdf_to_df()
# use df_plot() function to easily plot the output data frames:
```

```
df_plot(df)
df_plot(df2)
```

`tibble_to_vector` *Return a character vector from a tibble with one column.*

Description

`tibble_to_vector` takes a tibble with one column and returns the values in that column as a character vector.

Usage

```
tibble_to_vector(tib)
```

Arguments

`tib`	A tibble with only one column.

Details

This is a helper function for `pull_raw_pur`, `pull_clean_pur`, and `pur_filt_df`.

Value

A character vector.

Examples

```
library(magrittr)
tibble::tibble(x = 1:3) %>% tibble_to_vector()
```

`write_exposure` *Write exposure values for specified locations and dates in a certain county.*

Description

For a particular combination of locations and dates, radii, and active ingredients, `write_exposure` calculates exposure (kg/m^2) and writes out files (exposure values, meta data, and plots) to a specified directory.

Usage

```
write_exposure(clean_pur_df, locations_dates_df, radii, directory,
  chemicals = "all", aerial_ground = FALSE, write_plots = TRUE,
  verbose = TRUE, ...)
```

Arguments

clean_pur_df	A data frame returned by pull_clean_pur that includes data for the county of your locations, the time periods, and the active ingredients or cheimcal classes for which you want to calculate exposure.
locations_dates_df	A data frame with three columns: location, character strings of California addresses with street names, cities, state, and 5-digit zip codes, or pairs of coordinates in the form "longitude, latitude". All of the locations should be in the same county. The other two columns are start_date and end_date, with "yyyy-mm-dd" character strings specifying the time period(s) over which you would like to calculate exposure for each location.
radii	A vector of numeric values greater than zero that give the radii in meters defining the buffers around your locations in which you would like to calculate exposure. For reference, the length and width of a PLS section is about 1,609 meters (1 mile). That of a township could range from about 9,656 to 11,265 meters (6-7 miles).
directory	A path to the directory where you would like exposure files to be written. This directory will be created if it doesn't already exist.
chemicals	Either "all" or "chemical_class". The default is "all", which will calculate exposure to the summed active ingredients present in the clean_pur_df data frame. Enter "chemical_class" to calculate exposure to each of the chemical classes present in the chemical_class column of your clean_pur_df data frame.
aerial_ground	TRUE / FALSE for whether you would like to incorporate aerial/ground application into exposure calculations. If aerial_ground = TRUE, there should be an aerial_ground column in the input clean_pur_df data frame. There will be a value of exposure calculated for each chemical ("all" or by chemical class) and for each method of application: aerial or ground. The default is FALSE.
write_plots	TRUE / FALSE for whether you would like to write out plots returned from plot_exposure for each combination of location, date, radius, chemical class, and aerial/ground application. Plots are saved in a subdirectory called "exposure_plots".
verbose	TRUE / FALSE for whether you would like a message to print out while the function is running. The default is TRUE.
...	Additional arguments passed on to plot_exposure.

Value

Two .rds files ("exposure_df" and "meta_data") and a subdirectory ("exposure_plots") with PNG files of plot_exposure plots:

exposure_df.rds A data frame with 10 columns: exposure, the exposure value in kg/m^2 for that combination of chemicals, date range, aerial/ground application, location, and radius, chemicals, either "all" or a chemical class present in the provided clean_pur_df data frame, start_date and end_date, aerial_ground, which will be NA unless the aerial_ground argument is set to TRUE, location, radius, longitude and latitude of each location, and any error_messages that were returned when calculating exposure.

meta_data.rds A list with as many elements as there are rows in the exposure_df.rds data frame. Each element is a data frame with meta data relevant to the exposure value in the corresponding row of the exposure_df data frame. For example, meta data for exposure_df[1,] is saved as meta_data[[1]], exposure_df[2,] saved as meta_data[[2]], and so on. Meta data

data frames have 13 columns: `pls`, with each PLS unit intersected by the specified buffer, `chemicals`, either a chemical class or "all" (indicating exposure was calculated for all active ingredients present in the clean_pur_df data frame), `percent`, the percent intersection of the PLS unit with the buffer, `kg`, kg of active ingredients applied in the PLS unit for the given date range/chemicals/aerial_ground combination, `kg_intersection`, `percent` multiplied by `kg`, `start_date`, `end_date`, `aerial_ground` (this will be `NA` if the `aerial_ground` argument is set to `FALSE` or if `none_recorded` is `TRUE`), `none_recorded`, a logical value indicating if there were no active ingredients recorded for the PLS/date range/chemicals combination, `location`, `radius`, `area`, the area of the specified buffer, and `error_message`, which gives the error message, if any, that was returned.

exposure_plots A subdirectory with a `plot_exposure` plot saved for each row of the exposure_df data frame and element of the meta_data list. Plots are saved as #_exposure_plot.png, with numbers corresponding to the row number and element number of the exposure_df data frame and meta_data list, respectively. For example, 12_exposure_plot.png corresponds to exposure_df[12,] and meta_data[[12]].

Note

- Unlike the `calculate_exposure` function, `write_exposure` requires that you specify at least one start and end date for each location with the `locations_dates_df` argument. This is to accomodate multiple date ranges within a single location and differing start/end dates across locations.

Examples

```
library(magrittr)

chemical_class_df <- rbind(find_chemical_codes(2000:2001, "sulfur"),
                           find_chemical_codes(2000:2001, "methyl bromide")) %>%
   dplyr::rename(chemical_class = chemical)
pur <- pull_clean_pur(2000:2001, counties = "fresno",
                      chemicals = chemical_class_df$chemname,
                      sum_application = TRUE,
                      sum = "chemical_class",
                      chemical_class = chemical_class_df)
schools <- c("3333 American Ave., Fresno, CA 93725",
             "1111 Van Ness Ave., Fresno, CA 93721",
             "1616 South Fruit Ave., Fresno, CA 93706")
df <- data.frame(location = rep(schools, each = 2),
                 start_date = rep(c("2000-01-01", "2000-05-25", "2001-02-16"),
                                  each = 2),
                 end_date = c("2000-04-01", "2000-07-01",
                              "2000-08-25", "2000-11-25",
                              "2001-05-16", "2001-08-16"))
temp_dir <- tempdir()
write_exposure(pur, df, c(1500, 3000), "chemical_class",
               directory = temp_dir)
exposure_df <- readRDS(paste0(temp_dir, "/exposure_df.rds"))
meta_data <- readRDS(paste0(temp_dir, "/meta_data.rds"))
list.files(paste0(temp_dir, "/exposure_plots"))


spdf <- readRDS(system.file("extdata", "fresno_spdf.rds", package = "purexposure"))
pur <- readRDS(system.file("extdata", "fresno_clean.rds", package = "purexposure"))
df <- data.frame(location = "-119.726751, 36.660967",
```

```
                start_date = "2000-01-01",
                end_date = "2000-12-31")
temp <- tempdir()
write_exposure(pur, df, 3000, temp, spdf = spdf)
```

Index

∗Topic **datasets**
    california_shp, 5
    chemical_list, 6
    county_codes, 6

calculate_exposure, 2
california_shp, 5
chemical_list, 6
county_codes, 6

df_plot, 7

find_chemical_codes, 8
find_counties, 9
find_location_county, 9
find_product_name, 10

plot_application_timeseries, 11
plot_county_application, 12
plot_county_locations, 14
plot_exposure, 15
plot_locations_exposure, 17
pull_clean_pur, 18
pull_product_table, 22
pull_raw_pur, 23
pull_spdf, 24
purexposure, 25
purexposure-package *(*purexposure*)*, 25

spdf_to_df, 26

tibble_to_vector, 27

write_exposure, 27