Package 'purexposure'

November 6, 2017

Title Pull and Calculate Exposure to CA Pesticide Use Registry Records

Version 0.0.0.9000

Description Pulls and cleans California Pesticide Use Report (PUR) datasets data to visualize and calculate exposure to active ingredients present in applied pesticides.

Depends R (>= 3.4.1)

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),
rgdal (>= 1.2.8),
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 (>= 0.1.2.9000),
rgeos (>= 0.3.23),
magrittr (>= 1.5),
utils (>= 3.4.1),
colormap (>= 0.1.9000),
grDevices (<= 3.4.1),
methods (>= 3.4.1),
zoo (>= 1.8.0)

RoxygenNote 6.0.1

Suggests testthat,
knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

calculate_exposure . . . 2
california_shp . . . 4
chemical_list . . . 5
county_codes . . . 5
df_plot . . . 6
euc_distance . . . 7
find_chemical_codes . . . 7
find_counties . . . 8
find_location_county . . . 9
find_product_name . . . 9
gradient_n_pal2 . . . 11
help_calculate_exposure . . . 11
help_calc_exp . . . 12
help_categorize . . . 13
help_filter_pls . . . 14
help_find_chemical . . . 15
help_find_code . . . 15
help_find_product . . . 16
help_map_exp . . . 16
help_read_in_counties . . . 18
help_remove_cols . . . 18
help_return_exposure . . . 19
help_sum_ai . . . 20
help_sum_application . . . 20
help_write_md . . . 21
plot_application_timeseries . . . 22
plot_county_application . . . 23
plot_county_locations . . . 25
plot_exposure . . . 26
pull_clean_pur . . . 28
pull_product_table . . . 31
pull_pur_file . . . 33
pull_raw_pur . . . 34
pull_spdf . . . 35
purexposure . . . 36
scale_fill_gradientn2 . . . 37
spdf_to_df . . . 37
tibble_to_vector . . . 38

Index **39**

`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 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. For example, if you enter "6 months" for time_period, calculate_exposure will calculate exposure for every six month period starting from the earliest date 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 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.

Value

A list with five elements:

exposure A data frame with 7 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, and radius, the radius in meters for the buffer extending from 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 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 map_* or 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 calcualted.

Note

- If the `time_period`, `start_date`, and `end_date` arguments are all left as NULL (their defaults), then exposure will be estiamted 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 to take a look at the `county_plot` plot 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

```
## Not run:
clean_pur <- pull_clean_pur(2000, counties = "10")
exposure_list <- calculate_exposure(clean_pur,
                                    location = "13883 Lassen Ave, Helm, CA 93627",
                                    radius = 3000)
exposure_list$county_plot

# 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)
clean_pur3 <- pull_clean_pur(1995, "fresno",
                             chemicals = chemical_class_df$chemname,
                             sum_application = TRUE,
                             sum = "chemical_class",
                             chemical_class = chemical_class_df)
exp_list4 <- calculate_exposure(clean_pur3,
                                location = "13883 Lassen Ave, Helm, CA 93627",
                                radius = 1500,
                                chemicals = "chemical_class")
exp_list4$meta_data

# incorporate aerial/ground application information
clean_pur4 <- pull_clean_pur(2000, "yolo")
exp_list5 <- calculate_exposure(clean_pur4,
                                location = "-121.9018, 38.7646",
                                radius = 2500,
                                aerial_ground = TRUE)
exp_list5$exposure

## End(Not run)
```

`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 codes used to identify counties in California Pesticide Use Reports. This file, "county.txt", was pulled from the .zip file "pur2000.zip" found here: ftp://transfer.cdpr.ca.gov/pub/outgoing/pur_archives

Usage

```
county_codes
```

Format

A data frame with 58 rows and two columns:

county_name A character vector giving California county names.

county_code A character vector giving 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.

Source

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

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

```
## Not run:
shp <- pull_spdf("san diego", "township")
df <- spdf_to_df(shp)
df_plot(df)

## End(Not run)
```

euc_distance *Calculate euclidean distance between two points.*

Description

euc_distance calculates the straight-line distance between two points.

Usage

```
euc_distance(long, lat, origin_long, origin_lat)
```

Arguments

long	Longitude (x) of second point
lat	Latitude (y) of second point
origin_long	Longitude (x) of first point
origin_lat	Latitude (y) of first point

Details

This is a helper function for `calculate_exposure`.

Value

A data frame with one row and three columns: `long` and `lat` give the second point's coordinates, and `dist` gives the euclidian distance from these coordinates from the origin.

`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(year, chemicals = "all")
```

Arguments

`year`	A four-digit numeric year in the range of 1990 to 2015. Indicates the year 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.

Value

A data frame with three columns:

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

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.

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 = "codes")
```

Arguments

`return`	Either "codes" to return county codes (the default) or "names" to return county names.
`county`	A vector of character strings giving either a county names or two digit PUR county codes. Not case sensitive. California names and county codes as they appear in PUR datasets can be found in the `county_codes` dataset available with this package.

Value

If `return = "codes"`, a vector of two-character strings giving the corresponding PUR county codes. If `return = "names"`, a vector of county names.

Examples

```
find_counties(c("01", "03", "el dorado"))
find_counties(c("contra costa", "45"))

find_counties(c("01", "03", "el dorado"), return = "names")
find_counties(c("contra costa", "45"), return = "names")
find_counties("fresno")
```

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(location, return = "name", latlon_out = NULL)
```

Arguments

location	A length-one character string. Either a California address including street name, city, state, and zip code, or a pair of coordinates in the form "longitude, latitude".
return	Either "name" to return county name (the default) or "code" to return county code.
latlon_out	A numeric vector of two with longitude and latitude values. If the geocode code has been run earlier and this output is available, this saves a redundant request to the Google Maps API.

Value

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

Examples

```
## Not run:
address <- "13883 Lassen Ave, Helm, CA 93627"
find_location_county(location = address)

long_lat <- c("-120.09789, 36.53379")
find_location_county(location = long_lat)

## End(Not run)
```

---

find_product_name *Find Pesticide Product names and registration numbers from PUR Product Lookup Tables.*

Description

For a year and vector of 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(year, products = "all", download_progress = FALSE)
```

Arguments

year	A four digit year 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.
download_progress	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.

Value

A data frame with six columns:

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

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 The year for which product table information was pulled.

product Product name search terms.

Examples

```
## Not run:
prod_df <- find_product_name(2000, "mosquito")
prod_df <- find_product_name(2010, c("insecticide", "rodenticide"))

## End(Not run)
```

---

gradient_n_pal2 *Include alpha option in scales::gradient_n_pal().*

Description

This function adds an "alpha" argument from gradient_n_pal() from the scales package.

Usage

```
gradient_n_pal2(colours, values = NULL, space = "Lab", alpha = NULL)
```

help_calculate_exposure *Return exposure data for a single start and end date.*

Description

For a single date range, help_calculate_exposure returns the exposure and meta_data data frames to be output in the calculate_exposure list as a nested data frame.

Usage

```
help_calculate_exposure(start_date, end_date, aerial_ground, chemicals,
  clean_pur_df, location, pls_percents, pur_filt, radius)
```

Arguments

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.
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.
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.
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 zip code, or a pair of coordinates in the form "longitude, latitude".
pls_percents	A data frame
pur_filt	A data frame
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).

Details

This is a helper function for calculate_exposure.

Value

A nested data frame with two columns: The row_out column contains the exposure data frame for the date range, and meta_data contains the meta_data data frame for the date range.

help_calc_exp *Return a single exposure value for each combination of conditions.*

Description

help_calc_exp returns a data frame with exposure values (kg/m^2) by chemicals (including chemicals = "all") or by chemicals and aerial/ground application.

Usage

```
help_calc_exp(exp, buffer_area, ...)
```

Arguments

exp	A data frame
...	Either chemicals or chemicals, aerial_ground. Not quoted

Details

This is a helper function for help_calculate_exposure.

Value

A data frame with exposure values in kg/m^2 at a location for each relevant condition.

help_categorize *Categorize a continuous scale by percentile cutpoints.*

Description

Given a data frame with a column with a continuous numeric variable, help_categorize returns the data frame with a new column categorizing that continuous variable by percentile cutpoints.

Usage

```
help_categorize(section_data, buffer_or_county, start_date = NULL,
  end_date = NULL, aerial_ground = NULL, chemicals = NULL,
  clean_pur = NULL, s_t = NULL, percentile)
```

Arguments

section_data	A data frame with a continuous numeric variable named "kg"
buffer_or_county	Should cutpoints be determined by "county" or "buffer"?
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.

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.
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.
clean_pur	A pull_clean_pur data frame
s_t	"section" or "township"
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.

Details

This is a helper function for help_map_exp.

Value

The input section_data data frame with an additional column named category.

---

help_filter_pls *Find PLS units intersecting with a buffer.*

Description

help_filter_pls filters a SpatialPolygonsDataFrame to include only PLS units intersecting with a buffer, and filters the data frame returned from pull_clean_pur to include only those sections or townships.

Usage

```
help_filter_pls(pls, pls_quote, which_pls, shp, buffer, df, clean_pur_df)
```

Arguments

pls	Either MTRS (sections) or MTR (townships). Not quoted
pls_quote	Either "MTRS" or "MTR"
which_pls	A vector of character string PLS units
shp	A county's shapefile
buffer	A data frame with buffer coordinates
df	A data frame
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.

Details

This is a helper function for `calculate_exposure`.

Value

A list with four elements:

pur_filt A cleaned PUR data frame filtered to PLS units intersecting with a buffer.

comb_df_filt A spatial data frame with intersecting PLS units and a buffer.

pls_intersections A data frame with two columns: `pls` and `percent`, the corresponding percent intersection with the buffer.

pls_int A character vector with all PLS units intersecting with the buffer.

`help_find_chemical` *Pull a chemical code based on its name.*

Description

This function uses grep to return `chemname` values that match an input search term.

Usage

```
help_find_chemical(chemical, df)
```

Arguments

`chemical`	A string giving search term of a chemical to match with active ingredients present in pesticides applied in the given year.
`df`	A chemical table data frame.

Details

This is a helper function for `find_chemical_codes`.

Value

A data frame with three columns: `chem_code`, `chemname`, and `chemical`.

help_find_code *Find California county code or name*

Description

Given a county, help_find_code returns either the PUR county code or name.

Usage

```
help_find_code(county, return = "codes")
```

Arguments

county	A character string giving either a county name or two digit PUR county code. Not case sensitive. California names and county codes as they appear in PUR datasets can be found in the county_codes dataset available with this package.
return	Either "codes" to return county code (the default) or "names" to return county name.

Details

This is a helper function for find_counties.

Value

If return = "codes", a two-character string giving the corresponding PUR county codes. If return = "names", a county name.

Examples

```
help_find_code("01", return = "names")
help_find_code("contra costa", return = "codes")
```

help_find_product *Pull a chemical code based on its name.*

Description

This function uses grep to return chemname values that match an input search term.

Usage

```
help_find_product(product, df)
```

Arguments

df	A chemical table data frame.
chemical	A string giving search term of a chemical to match with active ingredients present in pesticides applied in the given year.

Details

This is a helper function for `find_chemical_codes`.

Value

A data frame with three columns: `chem_code`, `chemname`, and `chemical`.

`help_map_exp` *Return a map for a given exposure estimate.*

Description

For a unique combination of time periods, chemicals, and application methods, `help_map_exp` returns a plot showing amounts of pesticides applied for PLS units intersecting with the buffer.

Usage

```
help_map_exp(start_date, end_date, chemicals, aerial_ground, none_recorded,
  data_pls, gradient, location_longitude, location_latitude, buffer_df, buffer2,
  buffer, buffer_or_county, alpha, clean_pur, pls_labels, pls_labels_size,
  percentile, color_by)
```

Arguments

`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.
`data_pls`	A data frame
`gradient`	A character vector of hex color codes
`location_longitude`	A numeric longitude value
`location_latitude`	A numeric latitude value
`buffer_df`	A data frame
`buffer2`	A data frame
`buffer`	A gpc.poly object
`buffer_or_county`	"buffer" or "county"

alpha	A number in [0,1] specifying the transperency of fill colors. Numbers closer to 0 will result in more transparency. The default is 0.7.
clean_pur	A clean_pur_df data frame
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.
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.
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 dataset and date range. You can specify percentile cutpoints with the percentile argument.

Details

This is a helper function for plot_exposure.

Value

A list with three elements:

plot An exposure plot

data A data frame with information about each PLS unit

cutoff_values A data frame with cutoff values returned if color_by = "percentile"

---

help_read_in_counties *Read in PUR county dataset for a year.*

Description

Once a year of PUR data has been downloaded, this function reads in a selected county's dataset.

Usage

```
help_read_in_counties(code_or_file, type, year)
```

Arguments

code_or_file	A PUR county code or a file name for a county's dataset.
type	Either "codes" or "files", specifying the type of argument supplied to code_or_file.

Details

This is a helper function for pull_pur_file.

Value

A data frame of raw PUR data for a single county and year.

help_remove_cols *Remove columns with all missing values.*

Description

Given a quoted column name and its data frame, help_remove_cols determines if that column has all missing values or not.

Usage

```
help_remove_cols(col_quote, df)
```

Arguments

col_quote	A quoted column name
df	A data frame

Details

This is a helper function for help_sum_application.

Value

A data frame with two columns: col gives the column name, and all_missing is a logical value.

help_return_exposure *Return a data frame with exposure values and other related data.*

Description

For a single date range, help_return_exposure returns a data frame with exposure values calculated from help_calc_exp as well as other relevant data. This one row data frame is combined with data frames for other date ranges and then returned as the exposure element from a calculate_exposure list.

Usage

```
help_return_exposure(start_date, end_date, location, radius, exp, buffer_area,
  ...)
```

Arguments

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.
location	A length-one character string. Either a California address including street name, city, state, and 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).
exp	A data frame
...	Either chemicals or chemicals, aerial_ground. Not quoted.

Details

This is a helper function for help_calculate_exposure.

Value

A data frame with one row and the columns found in the calculate_exposure$exposure data frame.

---

help_sum_ai *Find summed applied active ingredients.*

Description

help_sum_ai finds the summed amount of applied active ingredients by section or township, chemical class, and aerial/ground application.

Usage

```
help_sum_ai(pur_filt, start_date, end_date, ...)
```

Arguments

pur_filt	A data frame
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.
...	A list of variables to group by. Options include section, township, chemical_class, and aerial_ground. Not quoted

Details

This is a helper function for help_calculate_exposure.

Value

A data frame a kg column and one to three additional columns, depending on the grouping variables.

help_sum_application *Sum application by section, township, chemical, and method of application.*

Description

help_sum_application sums application of a PUR dataset by chemicals, PLS unit, and aerial/ground application.

Usage

```
help_sum_application(df, sum, unit, aerial_ground, section_townships,
  chemical_class = NULL, ...)
```

Arguments

df	A data frame from pull_clean_pur before summing has taken place
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.
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.
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 FALSE.
section_townships	A section_townships data frame
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 dataset 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.
...	grouping variables

Details

This is a helper function for pull_clean_pur.

Value

A data frame. The number of columns is dependent on the grouping variables supplied to the ... argument.

help_write_md *Return a* meta_data *data frame.*

Description

help_write_md returns a data frame to be output as the meta_data element in the list returned from calculate_exposure.

Usage

```
help_write_md(clean_pur_df, pls_percents, pur_out, location, start_date,
  end_date, radius, buffer_area, mtrs_mtr, section_township)
```

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.
pls_percents	A data frame
pur_out	A data frame
location	A length-one character string. Either a California address including street name, city, state, and zip code, or a pair of coordinates in the form "longitude, latitude".
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.
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).
buffer_area	A numeric value
mtrs_mtr	Either MTRS or MTR. Not quoted
section_township	Either section or township. Not quoted

Details

This is a helper function for help_calculate_exposure.

Value

A data frame with the twelve columns in the calculate_exposure$meta_data data frame.

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, y_axis = "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 unqiue chemname or chemical_class column values. If facet = FALSE (the default), all active ingredients present in the dataset will be summed per day.
y_axis	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

```
## Not run:
pull_clean_pur(1990:1992, "fresno") %>%
    dplyr::filter(chemname %in% toupper(c("methyl bromide", "sulfur"))) %>%
    plot_application_timeseries(facet = TRUE)

pull_clean_pur(2000, "riverside") %>% plot_application_timeseries()

## End(Not run)
```

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_option = "viridis",
  crop = FALSE, alpha = 1)
```

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 datasets can be found in the county_codes dataset 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. 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 dataset 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 ingredents 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_option	A palette from the colormap package. The default is "viridis". See colormap palette options by visiting https://bhaskarvk.github.io/colormap/ or by running 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 transperency of fill colors. Numbers closer to 0 will result in more transparency. The default is 1.

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

```
## Not run:
tulare_list <- pull_clean_pur(2010, "tulare") %>% plot_county_application()

# plot all active ingredients
pur_df <- pull_clean_pur(2000:2001, "fresno", verbose = F)
fresno_list <- plot_county_application(pur_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

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

# map 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

## End(Not run)
```

`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`, or `county_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, one_plot = TRUE, fill_color = "red",
  alpha = 0.5)
```

Arguments

counties_or_df	A character vector of county names or county codes. You can use the county_codes dataset 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, or county_code column. If you provide a data frame, a plot for every county with data in that dataset will be output.
one_plot	TRUE / FALSE. If you provided multiple counties, whether you would like county outlines plotted in the same plot (TRUE), or if you would like separate plots returned in a list (FALSE). The default is TRUE
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.

Value

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

Examples

```
## Not run:
plot_county_locations("fresno")

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

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

## End(Not run)
```

---

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 applicaiton 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_option = "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 dataset 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_option	A palette from the colormap package. The default is "viridis". See colormap palette options by visiting https://bhaskarvk.github.io/colormap/ or by running colormap::colormaps.
alpha	A number in [0,1] specifying the transperency 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 estiamte, 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

```
## Not run:
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$exposure
tulare_list$cutoff_values

# 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_option = "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 amounts instead of percentile categories
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$chemname,
                              sum_application = TRUE,
                              sum = "chemical_class",
                              chemical_class = chemical_df) %>%
     calculate_exposure(location = "-119.6122, 34.90635",
                        radius = 3000,
                        time_period = "1 year",
                        chemicals = "chemical_class") %>%
     plot_exposure("amount")
do.call("rbind", santa_maria$exposure)
santa_maria$maps[[1]]
santa_maria$maps[[2]]
santa_maria$maps[[3]]

# scale colors based on buffer or county
turk <- pull_clean_pur(1996, "fresno") %>%
     dplyr::filter(chemname == "PHOSPHORIC ACID") %>%
     calculate_exposure(location = "-120.218404, 36.1806",
                        radius = 1500)

plot_exposure(turk, buffer_or_county = "county")$maps
plot_exposure(turk, buffer_or_county = "buffer")$maps

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

```
## End(Not run)
```

`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,
  download_progress = TRUE, raw_pur_df = NULL)
```

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 datasets. `years == "all"` will pull data from 1990 through 2015.
`counties`	A vector of character strings giving either a county name or a two digit county code for each county. Not case sensitive. California names and county codes as they appear in PUR datasets can be found in the `county_codes` dataset available with this package. For example, to return data for Alameda county, enter either "alameda" or "01" for the `counties` argument. `counties = "all"` will return data for all 58 California counties.
`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` dataset 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 FALSE.

`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.

`download_progress` 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 TRUE.

`raw_pur_df` A raw PUR data frame. Optional. If you've already downloaded a raw PUR data frame using `pull_raw_pur`, this argument prevents `pull_clean_pur` from downloading the same data again.

Value

A data frame with 13 columns:

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 7 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.

county_code A string two characters long giving the PUR 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 identifing 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. Will have values of `NA` if `sum_appliction = TRUE`.

outlier A logical value indicating whether the amount listed in `kg_chm_used` has been corrected large amounts entered in error. 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. Will have values of `NA` 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` dataset 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, check your working directory. You may want to change it back from a temporary directory.

Examples

```
## Not run:
df <- pull_clean_pur(download_progress = TRUE)
df2 <- pull_clean_pur(years = 2000:2010,
                      counties = c("01", "nevada", "riverside"),
                      chemicals = "methylene",
                      aerial_ground = TRUE)

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

df3 <- pull_clean_pur(2003) %>%
    dplyr::filter(prodno %in% prod_nos)

# Sum application by active ingredients
df4 <- pull_clean_pur(years = 2000:2010,
                      counties = c("01", "nevada", "riverside"),
                      chemicals = "methylene",
                      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)

df5 <- 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)

# clean an existing raw PUR dataset
placer_05 <- pull_raw_pur(2005, "placer")
df6 <- pull_clean_pur(raw_pur_df = placer_05)

## End(Not run)
```

`pull_product_table` *Pull PUR Product Table.*

Description

This function pulls a California Department of Pesticide Regulation Product Table for a particular year.

Usage

```
pull_product_table(year, download_progress = FALSE)
```

Arguments

`year`	A four digit year in the range of 1990 to 2015.
`download_progress`	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.

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 dataset 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

```
## Not run:
prod_95 <- pull_product_table(1995)

## End(Not run)
```

`pull_pur_file` *Pull raw PUR file for a single year and a county or counties.*

Description

`pull_pur_file` pulls the raw PUR dataset for a particular year and saves datasets for specified counties in a data frame.

Usage

```
pull_pur_file(year, counties = "all", download_progress = TRUE)
```

Arguments

`counties`	A character vector giving either county names or two digit county codes. Not case sensitive. California names and county codes as they appear in PUR datasets can be found in the county_codes dataset available with this package. For example, to return data for Alameda county, enter either "alameda" or "01" for the county argument. `counties = "all"` will pull data for all 58 California counties.
`download_progress`	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 TRUE.

Value

A data frame with 33 columns. Counties are indicated by `county_cd`; the year for which data was pulled is indicated by `applic_dt`.

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

```
## Not run:
raw\_file <- pull\_pur\_file(1999, c("40", "ventura", "yuba"))
raw\_file <- pull\_pur\_file(2015, "all")

## End(Not run)
```

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

Description

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

Usage

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

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 datasets. `years == "all"` will pull data from 1990 through 2015.
`counties`	A vector of character strings giving either a county name or a two digit county code for each county. Not case sensitive. California names and county codes as they appear in PUR datasets can be found in the `county_codes` dataset available with this package. For example, to return data for Alameda county, enter either "alameda" or "01" for the `counties` argument. `counties = "all"` will return data for all 58 California counties.
`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.
`download_progress`	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 TRUE.

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 want to change it back from a temporary directory.

Examples

```
## Not run:
df <- pull_raw_pur(download_progress = TRUE) # this will take a while to run
df2 <- pull_raw_pur(years = c(2000, 2010), counties = c("butte", "15", "01"))
df3 <- pull_raw_pur(years = 2015, counties = c("colusa"))

## End(Not run)
```

---

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", download_progress = FALSE)
```

Arguments

county	A vector of character strings giving either a county names or two digit PUR county codes. Not case sensitive. California names and county codes as they appear in PUR datasets can be found in the county_codes dataset 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.
download_progress	TRUE / FALSE indicating whether you would like a message and progress bar printed for the shapefile that is downloaded. The default value is FALSE

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

```
## Not run:
trinity_shp <- pull_spdf("trinity", download_progress = TRUE)
plot(trinity_shp)

del_norte_shp <- pull_spdf("08", "township", download_progress = TRUE)
plot(del_norte_shp)

## End(Not run)
```

`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. The main function categories are `find_*`, `pull_*`, `calculate_*`, and `plot_*`. These are the most important 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_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.

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_exposure`: Plot exposure to applied pesticides at a particular location.
- `plot_application_timeseries`: Plot time series of active ingredients in applied pesticides.

scale_fill_gradientn2 *Include alpha in ggplot2::scale_fill_gradientn().*

Description

This function adds an 'alpha' argument to scale_fill_gradientn() from the ggplot2 package.

Usage

```
scale_fill_gradientn2(..., colours, values = NULL, space = "Lab",
  na.value = "grey50", guide = "colourbar", colors, alpha = NULL)
```

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

```
## Not run:
df <- spdf_to_df(pull_spdf("fresno"))
df2 <- spdf_to_df(pull_spdf("sonoma"))

# use df_plot() function to easily plot the output data frames:
df_plot(df)
df_plot(df2)

## End(Not run)
```

`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.

Index

∗Topic **datasets**
    california_shp, 4
    chemical_list, 5
    county_codes, 5

calculate_exposure, 2
california_shp, 4
chemical_list, 5
county_codes, 5

df_plot, 6

euc_distance, 7

find_chemical_codes, 7
find_counties, 8
find_location_county, 9
find_product_name, 9

gradient_n_pal2, 11

help_calc_exp, 12
help_calculate_exposure, 11
help_categorize, 13
help_filter_pls, 14
help_find_chemical, 15
help_find_code, 15
help_find_product, 16
help_map_exp, 16
help_read_in_counties, 18
help_remove_cols, 18
help_return_exposure, 19
help_sum_ai, 20
help_sum_application, 20
help_write_md, 21

plot_application_timeseries, 22
plot_county_application, 23
plot_county_locations, 25
plot_exposure, 26
pull_clean_pur, 28
pull_product_table, 31
pull_pur_file, 33
pull_raw_pur, 34
pull_spdf, 35
purexposure, 36
purexposure-package *(*purexposure*)*, 36

scale_fill_gradientn2, 37
spdf_to_df, 37

tibble_to_vector, 38