Package 'qPRAentry'

May 9, 2026

Type Package

Title Quantitative Pest Risk Assessment at the Entry Step

Version 0.1.1

Description Supports risk assessors in performing the entry step of the quantitative Pest Risk Assessment. It allows the estimation of the amount of a plant pest entering a risk assessment area (in terms of founder populations) through the calculation of the imported commodities that could be potential pathways of pest entry, and the development of a pathway model. Two 'Shiny' apps based on the functionalities of the package are included, that simplify the process of assessing the risk of entry of plant pests. The approach is based on the work of the European Food Safety Authority (EFSA PLH Panel et al., 2018) <doi:10.2903/j.efsa.2018.5350>.

License GPL (>= 3)

URL https://github.com/mcendoya/qPRAentry

BugReports https://github.com/mcendoya/qPRAentry/issues

Depends R (>= 2.10)

Imports bsplus, dplyr, DT, eurostat (>= 4.0.0), ggiraph, ggplot2, giscoR (>= 0.6.0), memoise, purrr, sf, shiny, shinycssloaders, shinyjs, shinyWidgets, stats, tidyr

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Martina Cendoya [aut, cre] (ORCID: <https://orcid.org/0000-0002-0134-7781>), Maria Chiara Rosace [aut] (ORCID: <https://orcid.org/0000-0002-7263-8228>)

Maintainer Martina Cendoya <cendoya_marmar@gva.es>

Repository CRAN

Date/Publication 2025-04-09 20:10:02 UTC

Contents

`datatrade_EU` *Example Trade Data for the European Union*

Description

Simulated trade data for a commodity within the European Union (EU). This dataset illustrates the trade and production flow of a commodity that could serve as a potential pathway for the entry of a pest into EU countries, in yearly time periods. It also includes simulated consumption data for the commodity at NUTS1 level. This dataset serves as a reference for how the data should be structured and as a basis for the examples included in the qPRAentry package to deal with the NUTS code system (Nomenclature of territorial units for statistics).

Usage

```
datatrade_EU
```

Format

A list of four data frames:

- **extra_import** Data on imports from countries outside of the EU. A data frame with 216 rows and 4 columns:

`reporter`	NUTS0 code of the importing country.
`partner`	ID of the exporting countries, coded as "Extra_Total" for all external countries, and "CNTR_1", "CNTR_

time_period	Time period of trade, with values 2020 and 2021.
value	Quantity of the commodity imported.

- **intra_trade** Data on internal trade within EU member countries. A data frame with 1404 rows and 4 columns:

reporter	NUTS0 code of the importing country.
partner	NUTS0 code of the exporting country.
time_period	Time period of trade, with values 2020 and 2021.
value	Quantity of the commodity imported.

- **internal_production** Data on the internal production of the commodity within the EU. A data frame with 54 rows and 3 columns:

reporter	NUTS0 code of the producing country.
time_period	Time period of trade, in years (2020 and 2021).
value	Quantity of the commodity produced.

- **consumption_nuts1** Data on the consumption of the commodity in the EU at NUTS1 level (administrative divisions). A data frame with 92 rows and 2 columns:

NUTS_ID	NUTS1 code of the administrative division.
value	Quantity of the commodity consumed.

`datatrade_NorthAm` *Example Trade Data for Northern America*

Description

Simulated trade data for a commodity within Northern America. This dataset illustrates the trade and production flow of a commodity that could serve as a potential pathway for the entry of a pest into Northern American countries. The data is organised in three-month periods within a year. It also includes simulated consumption data of the commodity at at the level of principal sub-national divisions (ISO 3166-2 codes). This dataset serves as a reference on how the data should be structured and provides the basis for the examples included in the qPRAentry package to deal with the ISO 3166 code system (ISO 3166 Maintenance Agency).

Usage

```
datatrade_NorthAm
```

Format

A list of four data frames:

- **extra_import** Data on imports from countries outside of Northern America. A data frame with 100 rows and 4 columns:

reporter	ISO 3166-1 alpha-2 code of the importing country.
partner	ID of the exporting country, coded as "CNTR_1" to "CNTR_5" to represent specific non-Northern Ameri
time_period	Time period of trade, in three-month periods ("January-March", "April-June", "July-September", "Octobe
value	Quantity of the commodity imported.

- **intra_trade** Data on internal trade within Northern American countries. A data frame with 80 rows and 4 columns:

reporter	ISO 3166-1 alpha-2 code of the importing country.
partner	ISO 3166-1 alpha-2 code of the exporting country.
time_period	Time period of trade, in three-month periods ("January-March", "April-June", "July-September", "Octobe
value	Quantity of the commodity imported.

- **internal_production** Data on the internal production of the commodity within Northern America. A data frame with 20 rows and 3 columns:

reporter	ISO 3166-1 alpha-2 code of the producing country.
time_period	Time period of trade, in three-month periods ("January-March", "April-June", "July-September", "Octobe
value	Quantity of the commodity produced.

- **consumption_iso2** Data on the consumption of the commodity in Northern America at ISO 3166-2 level (principal subdivisions of countries). A data frame with 83 rows and 2 columns:

iso_3166_2	ISO 3166-2 code of the subdivision.
value	Quantity of the commodity consumed.

load_csv *Load a CSV file with automatic separator detection*

Description

Reads CSV files containing more than one column of data. It automatically detects common separators (i.e., comma, semicolon, tab), while allowing users to specify encoding and decimal separators as required.

Usage

```
load_csv(filepath, dec = ".", encoding = getOption("encoding"))
```

Arguments

filepath	A string specifying the path to the CSV file (.csv extension).
dec	A character specifying the decimal separator to use. Default is "." (period).
encoding	A string specifying the encoding of the file. Default is the native system encoding set by `getOption("encoding")`. For more details on encoding see file.

Value

A data frame containing the data from the CSV file.

Examples

```
# file path
fpath <- system.file("extdata", "data_ex.csv", package="qPRAentry")
# Load a CSV file
df <- load_csv(fpath)
head(df) # value as character
# Load a CSV file with comma separated decimals
df <- load_csv(fpath, dec = ",") # value as numeric
```

ntrade *Ntrade calculation*

Description

Calculates the quantity of potentially infested imported commodity (N_{trade}) from third countries where the pest is present, based on the provided trade data (`TradeData` object output of the `trade_data()` function).

Usage

```
ntrade(
  trade_data,
  filter_IDs = NULL,
  filter_period = NULL,
  summarise_result = NULL
)
```

Arguments

`trade_data`	An object of class `TradeData` that can be the output of `trade_data()`.
`filter_IDs`	A vector containing the country IDs to filter (identification codes of the countries of interest). By default, it is set to `NULL`, meaning all `reporter` countries in the data frames will be considered.
`filter_period`	A vector specifying the time periods to filter, based on the `time_period` column. By default, it is set to `NULL`, meaning all time periods in the data frames will be considered.
`summarise_result`	A character vector specifying functions to summarise the N_{trade} result for the selected time periods (`filter_period`). It accepts the expressions `"mean"` for the mean, `"sd"` for the standard deviation, `"median"` for the median value and `"quantile(p)"` where `p` is the probability for the quantiles to the given probabilities. See examples.

Details

The calculation of N_{trade_i} for each country of interest i is based on the equation:

$$N_{trade_i} = ExtraPest_i - ExtraPest_i \sum_{j \neq i} R_{ij} + \sum_{j \neq i} ExtraPest_j R_{ji},$$

where:

- N_{trade_i}: quantity of commodity from third countries remaining in country i, taking into account the direct importation from third countries where the pest is present, the re-exportation to other countries of interest, and the indirect importation of the commodity from other countries of interest.
- $ExtraPest_i$ and $ExtraPest_j$: quantity of commodity imported by country i and country j from third countries where the pest is present (direct import), during the period of time considered.
- R_{ij} and R_{ji}: proportion of intra-regional trade relative to the total available quantity in the exporting country defined as:

  $$R_{ij} = IntraExp_{ij}/(IP_i + ExtraTotal_i), R_{ji} = IntraExp_{ji}/(IP_j + ExtraTotal_j).$$

  Specifically, R_{ij} indicates the proportion of the commodity that is exported from country i to country j ($IntraExp_{ij}$), while R_{ji} indicates the proportion exported from country j to country i ($IntraExp_{ji}$), in both cases out of the total available commodity in the exporter

country. The total available quantity is considered as tha sum of the internal production of the country (IP) and the total quantity imported from third countries ($ExtraTotal$). Thus, the quantity of $ExtraPest_i$ re-exported from country i to all countries j is approximated by $ExtraPest_i \sum_{j \neq i} R_{ij}$, and the quantity of $ExtraPest_j$ re-exported from all countries j to country i as $\sum_{j \neq i} ExtraPest_j R_{ji}$.

Value

A data frame with the quantity of commodity imported by each country of interest (`country_IDs`) from countries or regions where the pest is present. The result is returned for each time period if `summarise_result` is not specified (default is `NULL`). If a summary function is specified, the result will be summarised accordingly.

See Also

`trade_data()`

Examples

```
## Example with simulated trade data for Northern America
library(dplyr)
data("datatrade_NorthAm")
# Total extra-import data: data contains imports from 5 third countries (column partner).
extra_total <- datatrade_NorthAm$extra_import
# Extra-import data from countries where the pest is present (e.g., CNTR_1 and CNTR_2)
CNTR_pest <- c("CNTR_1", "CNTR_2")
extra_pest <- datatrade_NorthAm$extra_import %>% filter(partner%in%CNTR_pest)
# Intra-trade data
intra_trade  <- datatrade_NorthAm$intra_trade
# Internal production data
internal_production  <- datatrade_NorthAm$internal_production
# Generate trade data (TradeData object)
trade_NorthAm <- trade_data(extra_total = extra_total,
                            extra_pest = extra_pest,
                            intra_trade = intra_trade,
                            internal_production = internal_production)
# Calculation of the Ntrade for each time period
ntrade_NorthAm <- ntrade(trade_data = trade_NorthAm)
head(ntrade_NorthAm)
# Ntrade summary for the time periods
ntrade_NorthAm_summary <- ntrade(trade_data = trade_NorthAm,
                                 summarise_result = c("mean", "sd",
                                                      "quantile(0.025)",
                                                      "median",
                                                      "quantile(0.975)"))
head(ntrade_NorthAm_summary)
# Plot the median of Ntrade
library(ggplot2)
plot_countries(data = ntrade_NorthAm_summary,
               iso_col = "country_IDs",
               values_col = "median") +
  xlim(-180,-20) + ylim(0,90)
```

```
## Example with simulated trade data for Europe
# Load data
data("datatrade_EU")
# Total extra-import data: the total import is identified as partner "Extra_Total"
extra_total <- datatrade_EU$extra_import %>% filter(partner=="Extra_Total")
# Extra-import data from countries where the pest is present
extra_pest <- datatrade_EU$extra_import %>% filter(partner!="Extra_Total")
# Intra-trade data
intra_trade  <- datatrade_EU$intra_trade
# Internal production data
internal_production  <- datatrade_EU$internal_production
# Generate trade data (TradeData object)
trade_EU <- trade_data(extra_total = extra_total,
                       extra_pest = extra_pest,
                       intra_trade = intra_trade,
                       internal_production = internal_production)
# Ntrade mean and sd for the time periods
ntrade_EU <- ntrade(trade_data = trade_EU,
                    summarise_result = c("mean", "sd"))
# Plot Ntrade mean
plot_countries(data = ntrade_EU,
               iso_col="country_IDs",
               values_col="mean") +
  xlim(-40,50) + ylim(25,70)
# Ntrade for selected countries and a specific time period
# Sample 5 countries from trade data
country_IDs <- sample(unique(trade_EU$total_trade$country_IDs), 5)
ntrade_EU_s <- ntrade(trade_data = trade_EU,
                      filter_IDs = country_IDs,
                      filter_period = 2020)
head(ntrade_EU_s)
# Plot Ntrade result
plot_countries(data = ntrade_EU_s,
               iso_col="country_IDs",
               values_col="Ntrade_2020") +
  xlim(-40,50) + ylim(25,70)
```

`ntrade_app` *Shiny app for Ntrade*

Description

Interactive application for the calculation and redistribution of the potentially infected/infested quantity of commodities imported by a country from third countries where the pest is present (N_{trade}) using the NUTS coding system. See Nomenclature of territorial units for statistics.

Usage

```
ntrade_app()
```

Value

No return value, called for side effects

See Also

`ntrade()`, `redist_nuts()`, `redist_iso()`

Examples

```
if (interactive()) {
ntrade_app()
}
```

`pathway_app` *Shiny app for pathway model*

Description

Interactive application to estimate the number of potential founder populations ($NPFP$) of a pest in different regions or countries using the NUTS coding system. See Nomenclature of territorial units for statistics.

Usage

```
pathway_app()
```

Value

No return value, called for side effects

See Also

`pathway_model()`

Examples

```
if (interactive()) {
pathway_app()
}
```

`pathway_model` *Pathway model*

Description

Estimates the number of potential founder populations ($NPFP$) of a pest in different regions, using N_{trade} data combined with additional user-defined parameters.

Usage

```
pathway_model(
  ntrade_data,
  IDs_col,
  values_col,
  expression,
  parameters,
  niter = 100
)
```

Arguments

`ntrade_data`	A data frame with the quantity of potentially infested commodities imported from third countries where the pest is present (N_{trade}). It can be calculated using the `ntrade()` function.
`IDs_col`	A string specifying the column name in `ntrade_data` with the country or region IDs of interest. See details on **IDs - country or region identification codes**.
`values_col`	A string specifying the column name in `ntrade_data` with the N_{trade} values (quantity of potentially infested commodity imports) to be used in the pathway model.
`expression`	A string of characters representing the equation for the pathway model. This expression must not include N_{trade}, since it is added multiplicatively to the entered equation by default. The resulting equation will be of the form: $$NPFP = N_{trade_i} \cdot \text{“}expression\text{”}$$
`parameters`	A named list specifying the distributions for each parameter used in `expression`. Each element of the list must be another list containing: • `dist`: A string indicating the distribution name (e.g., "norm", "beta"). • Additional arguments required by the specified distribution (e.g., `mean`, `sd` for "norm"). See details on **Parameter distributions** for a list of available distributions and examples on how to specify them.
`niter`	The number of iterations to generate random samples from the distributions. The default is 100 iterations.

Details

IDs - country or region identification codes::

The use of ISO 3166 (alpha-2) codes (ISO 3166 Maintenance Agency), or NUTS codes in the case of European countries Nomenclature of territorial units for statistics, as country or region identifiers (`IDs_col`) is recommended for subsequent compatibility with other functions of the qPRAentry package.

Parameter distributions::

The following distributions are supported. For details on their parameters, refer to the linked R documentation:

Distribution	Documentation
"beta"	`rbeta()` (Beta distribution)
"binom"	`rbinom()` (Binomial distribution)
"cauchy"	`rcauchy()` (Cauchy distribution)
"chisq"	`rchisq()` (Chi-squared distribution)
"exp"	`rexp()` (Exponential distribution)
"f"	`rf()` (F distribution)
"gamma"	`rgamma()` (Gamma distribution)
"geom"	`rgeom()` (Geometric distribution)
"lnorm"	`rlnorm()` (Log-normal distribution)
"nbinom"	`rnbinom()` (Negative Binomial distribution)
"norm"	`rnorm()` (Normal distribution)
"pois"	`rpois()` (Poisson distribution)
"t"	`rt()` (Student's t distribution)
"unif"	`runif()` (Uniform distribution)
"weibull"	`rweibull()` (Weibull distribution)

For example, to specify a normal distribution with mean 0 and standard deviation 1:
`list(dist = "norm", mean = 0, sd = 1)`

Ensure that all parameters required by the chosen distribution are included.

Value

A data frame with the statistics (mean, SD, minimum, first quartile, median, third quartile, and maximum) resulting from the iterations of the $NPFP$ for each country/region and for the total (i.e., the results for the set of all countries/regions).

See Also

`ntrade()`

Examples

```
## Example using Northern American countries and ntrade simulated data
data("datatrade_NorthAm")
# Extract country IDs and simulate ntrade data
IDs <- datatrade_NorthAm$internal_production$reporter
```

```
df <- data.frame(IDs = IDs,
                 ntrade_values = abs(rnorm(length(IDs), 10000, 2000)))
# Expression for the pathway model using 3 parameters
eq <- "(1/P1)*P2*P3"
# Distribution for each parameter
parameters <- list(
  P1 = list(dist = "beta", shape1 = 0.5, shape2 = 1),
  P2 = list(dist = "gamma", shape = 1.5, scale = 100),
  P3 = list(dist = "lnorm", mean = 5, sd = 2)
)
# Run pathway_model()
res_pathway <- pathway_model(ntrade_data = df,
                             IDs_col = "IDs",
                             values_col = "ntrade_values",
                             expression = eq,
                             parameters = parameters,
                             niter = 100)
head(res_pathway)
# summary of the total for all countries
res_pathway[res_pathway$IDs == "Total",]
# plot
plot_countries(res_pathway, "IDs", "Median")
```

`plot_countries` *Plot values on a map at country level*

Description

Plots country values on a map using data provided and allows customisation of various aesthetics, such as colors, legend title, and title.

Usage

```
plot_countries(
  data,
  iso_col,
  values_col,
  colors = NULL,
  na_value = "grey",
  title = NULL,
  legend_title = NULL
)
```

Arguments

`data`	A data frame containing the values to be plotted on the map.
`iso_col`	A string specifying the column name in `data` with the ISO 3166-1 (alpha-2) country codes. See ISO 3166 Maintenance Agency for details on country codes.

values_col	A string specifying the column name in data with the values to be plotted.
colors	Optional vector of colors used in the gradient scale.
na_value	Color for missing values (default is "grey").
title	A title for the plot (default is NULL).
legend_title	A title for the legend. Default NULL, name in the values_col.

Details

Extracts an sf object from the giscoR package. It uses the ggplot2 package for the representation. Also, it supports the addition of other ggplot2 options (see examples).

Value

A ggplot object with the plotted countries.

Examples

```
# Simulated data trade in Northern America
data("datatrade_NorthAm")
# Mean of internal production for each country
library(dplyr)
data_plot <- datatrade_NorthAm$internal_production %>%
  group_by(reporter) %>%
  summarise(mean_value = mean(value))

head(data_plot)

#Plot
pl <- plot_countries(data = data_plot,
                     iso_col = "reporter",
                     values_col = "mean_value")
pl

# Changing colors and adding other ggplot2 options
library(ggplot2)
pl <- plot_countries(data = data_plot,
                     iso_col = "reporter",
                     values_col = "mean_value",
                     colors = c("white", "lightblue", "darkblue"),
                     title = "Plot internal production",
                     legend_title = "units")
pl +
  xlim(-170, -20) + ylim(10, 90) +
  theme_bw()
```

plot_nuts *Plot NUTS region values on a map*

Description

Plots NUTS region values on a map using the provided data and allows customisation of various aesthetics, such as colors, legend title, and title.

Usage

```
plot_nuts(
  data,
  nuts_col,
  values_col,
  nuts_level = 2,
  nuts_year = "2016",
  colors = NULL,
  na_value = "grey",
  title = NULL,
  legend_title = NULL
)
```

Arguments

data	A data frame containing the values to be plotted on the map.
nuts_col	A string specifying the column name in data containing NUTS codes.
values_col	A string specifying the column name in data with the values to be plotted.
nuts_level	A numeric value (0, 1, 2, or 3) specifying the NUTS level to plot. Default is 2 indicating NUTS2. See Nomenclature of territorial units for statistics.
nuts_year	Year of NUTS classification. Accepted values are '2003','2006', '2010','2013','2016' (default),'2021', or '2024'. See NUTS - History.
colors	Optional vector of colors used in the gradient scale.
na_value	Color for missing values (default is "grey").
title	A title for the plot (default is NULL).
legend_title	A title for the legend. Default NULL, name in the values_col.

Details

Extracts an sf object from the giscoR package. It uses the ggplot2 package for the representation. Also, it supports the addition of other ggplot2 options (see examples).

Value

A ggplot object with the plotted NUTS regions.

Examples

```
## Example plot at NUTS0 level (country level)
# Simulated data trade in European countries
data("datatrade_EU")
# Mean of internal production for each country
library(dplyr)
data_plot <- datatrade_EU$internal_production %>%
  group_by(reporter) %>%
  summarise(mean_value = mean(value))

head(data_plot)

#Plot
pl <- plot_nuts(data = data_plot,
                nuts_col = "reporter",
                values_col = "mean_value",
                nuts_level = 0)
pl

## Example plot at NUTS1 level (codes extracted from 'giscoR' package)
library(dplyr)
library(giscoR)
data_plot <- gisco_get_nuts(nuts_level=1) %>%
  select(NUTS_ID) %>%
  # simulate values for each NUTS1
  mutate(values = abs(rnorm(nrow(.), 0, 1000)))

#Plot
pl <- plot_nuts(data = data_plot,
                nuts_col = "NUTS_ID",
                values_col = "values",
                nuts_level = 1,
                colors = c("white", "lightblue", "darkblue"),
                title = "NUTS1",
                legend_title = "units")

# Changing colors and adding other ggplot2 options
library(ggplot2)
pl +
  xlim(-40, 50) + ylim(20, 70) +
  theme_bw()
```

`redist_iso` *Data redistribution to country subdivisions*

Description

Value redistribution from country-level (i.e., ISO 3166-1 alpha-2 codes) to principal subdivisions (i.e., ISO 3166-2 codes). See ISO 3166 Maintenance Agency.

Usage

```
redist_iso(
  data,
  iso_col,
  values_col,
  redist_data,
  redist_iso_col,
  redist_values_col
)
```

Arguments

`data`	A data frame containing the data at the country-level to redistribute.
`iso_col`	A string specifying the column name in `data` with the ISO 3166-1 (alpha-2) country codes.
`values_col`	A string or vector specifying the column name(s) in `data` with the values to be redistributed.
`redist_data`	A data frame with values for each subdivision that will be used as the basis for redistribution.
`redist_iso_col`	A string specifying the column name in `redist_data` that contains the destination ISO 3166-2 codes.
`redist_values_col`	A string specifying the column name in `redist_data` with the values for proportional redistribution. This will define the weights used for the redistribution.

Details

This function enables redistribution of values from country-level to principal subdivisions (e.g., provinces or states), proportionally to user-supplied redistribution proportions.

Note that more than one column of values provided in the data frame data can be redistributed at the same time. The values in columns `values_col` and `redist_values_col` must be numeric and positive.

Common uses:

In the context of quantitative pest risk assessment (qPRA) at the entry step, this function can be applied to redistribute the quantity of potentially infested commodities (N_{trade}, see `ntrade()`) or the number of potential founder populations ($NPFP$, see `pathway_model()`). For this purpose, human population or consumption data from subdivisions are often used for redistribution.

Value

A data frame with the redistributed values across the specified subnational level. The data frame contains the columns `ISO_1` with the codes at country level, `ISO_2` with the codes at subdivision level, `proportion` with the proportion according to which the values have been redistributed, and the columns corresponding to the redistributed values with the same name specified in `values_col`.

See Also

ntrade(), pathway_model()

Examples

```
## Example of data redistribution in Northern American countries
data(datatrade_NorthAm)
# Selection of internal production data from January to March to be proportionally
# redistributed based on sub-national consumption data
data_ip <- datatrade_NorthAm$internal_production
data_ip <- data_ip[data_ip$time_period=="January-March",]
# consumption data at sub-national level
data_sub <- datatrade_NorthAm$consumption_iso2

# Redistribution
data_redist <- redist_iso(data = data_ip,
                          iso_col = "reporter",
                          values_col = "value",
                          redist_data = data_sub,
                          redist_iso_col = "iso_3166_2",
                          redist_values_col = "value")

head(data_redist)
```

---

redist_nuts *Data redistribution to NUTS subdivisions*

Description

Value redistribution from country-level (NUTS0) to smaller territories (i.e., NUTS1, NUTS2 or NUTS3). See Nomenclature of territorial units for statistics.

Usage

```
redist_nuts(
  data,
  nuts_col,
  values_col,
  to_nuts = 2,
  redist_data = "population",
  redist_nuts_col = NULL,
  redist_values_col = NULL,
  population_year = 2023,
  nuts_year = 2016
)
```

Arguments

`data`	A data frame containing the data at the country-level to redistribute.
`nuts_col`	A string specifying the column name in `data` with the NUTS0 codes.
`values_col`	A string or vector specifying the column name(s) in `data` with the values to be redistributed.
`to_nuts`	A numeric value (1, 2, or 3) specifying the NUTS level for redistribution. Default 2, indicating redistribution to NUTS2.
`redist_data`	A data frame containing values for each subdivision that will be used as the basis for proportional redistribution. By default, this is set to `"population"`, indicating redistribution based on human population data from Eurostat.
`redist_nuts_col`	A string specifying the column name in `redist_data` that contains the destination NUTS codes. The NUTS level should correspond to the value specified in `to_nuts`. `NULL` (default) if a data frame is not incorporated in `redist_data` (i.e., `redist_data = "population"`).
`redist_values_col`	A string specifying the column name in `redist_data` with the values for proportional redistribution. These values will serve as the weights for the redistribution process. `NULL` (default) if a data frame is not incorporated in `redist_data` (i.e., `redist_data = "population"`).
`population_year`	A numeric value specifying the year(s) of the human population data to be used for redistribution. Only necessary if `"population"` is specified in `redist_data` (default is 2023). If multiple years are provided, the average human population across those years will be used. Available years can be found at Eurostat.
`nuts_year`	Year of NUTS classification. Accepted values are '2003','2006','2010','2013', '2016' (default),'2021', or '2024'. See NUTS - History.

Details

This function enables redistribution of values from national-level NUTS0 to smaller territorial units (i.e., NUTS1, NUTS2, or NUTS3), either proportionally based on human population data from Eurostat or using user-supplied redistribution proportions. Human population data for redistribution is automatically fetched for the specified time period from the Eurostat database.

Note that more than one column of values provided in the data frame `data` can be redistributed at the same time. The values in columns `values_col` and `redist_values_col` must be numeric and positive.

Common uses:

In the context of quantitative pest risk assessment (qPRA) at the entry step, this function can be applied to redistribute the quantity of potentially infested commodities (N_{trade}, see `ntrade()`) or the number of potential founder populations ($NPFP$, see `pathway_model()`). For this purpose, human population or consumption data from subdivisions are often used for redistribution.

Value

A data frame with the redistributed values across the specified NUTS level. The data frame contains the columns `NUTSX` with the codes at the selected NUTS level, `NUTS0` with the codes at country level, `proportion` with the proportion according to which the values have been redistributed, and the columns corresponding to the redistributed values with the same name specified in `values_col`.

See Also

`ntrade()`, `pathway_model()`

Examples

```
## Example of data redistribution to NUTS2 using human population data
data("datatrade_EU")
# extract NUTS0 codes (country level)
nuts0 <- unique(datatrade_EU$internal_production$reporter)
# simulate values for each country
nuts0_data <- data.frame(nuts0 = nuts0,
                         value = abs(rnorm(length(nuts0), 30000, 10000)))


# Redistribution
data_redist <- redist_nuts(data = nuts0_data,
                           nuts_col = "nuts0",
                           values_col = "value",
                           to_nuts = 2,
                           redist_data = "population",
                           population_year = c(2017, 2018, 2019))
head(data_redist)

# Plot
plot_nuts(data = data_redist,
          nuts_level = 2,
          nuts_col = "NUTS2",
          values_col = "value")


## Example of data redistribution to NUTS1 using custom data
# consumption data at NUTS1 level
nuts1_data <- datatrade_EU$consumption_nuts1

# Redistribution
data_redist <- redist_nuts(data = nuts0_data,
                           nuts_col = "nuts0",
                           values_col = "value",
                           to_nuts = 1,
                           redist_data = nuts1_data,
                           redist_nuts_col = "NUTS_ID",
                           redist_values_col = "value")

head(data_redist)
```

```
# Plot
plot_nuts(data = data_redist,
          nuts_level = 1,
          nuts_col = "NUTS1",
          values_col = "value")
```

`trade_data` *Prepare Trade Data*

Description

Prepares trade data for each country of interest based on the provided data. Generates objects of class `TradeData` required to be used in the `ntrade()` function of the qPRAentry package.

Usage

```
trade_data(
  extra_total,
  extra_pest,
  intra_trade,
  internal_production,
  filter_IDs = NULL,
  filter_period = NULL
)
```

Arguments

`extra_total`	A data frame containing the total quantity of commodity imported from third countries (pest-free and pest-present countries). It must contain the following columns: `reporter` (importing country), `partner` (exporting country), `value` (quantity of commodity) and `time_period` (time period of the trade activity).
`extra_pest`	A data frame containing the quantity of commodity imported from third countries where the pest is present. It must contain the following columns: `reporter` (importing country), `partner` (exporting country), `value` (quantity of commodity) and `time_period` (time period of the trade activity). The quantity of imported commodity detailed in this data frame must also be included in the `extra_total` data frame.
`intra_trade`	A data frame containing the quantity of commodity traded between countries of interest. It must contain the following columns: `reporter` (importing country), `partner` (exporting country), `value` (quantity of commodity) and `time_period` (time period of the trade activity).
`internal_production`	A data frame containing the quantity of commodity produced internally within each country of interest. It must contain the following columns: `reporter` (producing country), `value` (quantity of commodity) and `time_period` (time period of the production).

`filter_IDs`	A vector containing the country IDs to filter (identification codes of the countries of interest). By default, it is set to `NULL`, meaning all `reporter` countries in the data frames will be considered.
`filter_period`	A vector specifying the time periods to filter, based on the `time_period` column. By default, it is set to `NULL`, meaning all time periods in the data frames will be considered.

Details

The function combines external imports from third countries, internal trade between the countries of interest and internal production data. It calculates the total amount of product available per country in each time period as the sum of external imports (from pest-free and pest-present countries) and internal production.

IDs - country identification codes::

For the IDs of the countries of interest (i.e., in the the columns `reporter` of the four trade data frames and in the column `partner` of `intra_trade`) it is recommended to use the the ISO 3166-1 (alpha-2) codes (ISO 3166 Maintenance Agency) or NUTS0 codes in case of European countries (Nomenclature of territorial units for statistics) for subsequent compatibility with other functions of the qPRAentry package.

Time periods::

Time periods can be specified in any way, both numeric and character formatting is supported. For example, it can be expressed as years, months, specific periods, seasons, etc.

Trade adjustments::

Trade imbalances are adjusted, so that in case the internal export for a given country exceeds the total quantity available in that country, the internal export is recalculated proportionally based on the total available. Missing values are treated as zeros.

Value

An object of class `TradeData` is returned containing the following list of data frames:

- `total_trade` A data frame with one row for each ID and each time period with 9 variables:

`country_IDs`	IDs of the countries of interest.
`time_period`	Time period.
`extra_total`	Total imports from third countries.
`extra_pest`	Imports from third countries where the pest of interest is present.
`intra_import`	Internal import from the countries of interest.
`intra_export`	Internal export to the countries of interest.
`internal_production`	Internal production in the countries of interest.

total_available	Total available quantity in the countries of interest.
export_prop	Proportion of internal export to the total available commodity. A value of 1 indicates that intern

- intra_trade A data frame with values of trade commodity between countries of interest:

reporter	Importing country ID.
partner	Exporting country ID.
time_period	Time period.
value	Quantity of the commodity traded.
export_prop	Proportion of internal export to the total available commodity for each trading partner according to the pr

See Also

load_csv(), ntrade()

Examples

```
## Example with simulated trade data for Northern America
library(dplyr)
# Load data
data("datatrade_NorthAm")
# Total extra-import data: data contains imports from 5 third countries (column partner).
extra_total <- datatrade_NorthAm$extra_import
# Extra-import data from countries where the pest is present (e.g., CNTR_1 and CNTR_2)
CNTR_pest <- c("CNTR_1", "CNTR_2")
extra_pest <- datatrade_NorthAm$extra_import %>% filter(partner%in%CNTR_pest)
# Intra-trade data
intra_trade  <- datatrade_NorthAm$intra_trade
# Internal production data
internal_production  <- datatrade_NorthAm$internal_production
# Generate trade data (TradeData object)
trade_NorthAm <- trade_data(extra_total = extra_total,
                            extra_pest = extra_pest,
                            intra_trade = intra_trade,
                            internal_production = internal_production)
head(trade_NorthAm$total_trade)
head(trade_NorthAm$intra_trade)
# Plot the total available quantity of commodity available in each country
library(ggplot2)
plot_countries(data = trade_NorthAm$total_trade,
               iso_col = "country_IDs",
               values_col = "total_available") +
  xlim(-180,-20) + ylim(0,90)
```

```
## Example with simulated trade data for Europe
# with selected countries and a specific time period
# Load data
data("datatrade_EU")
# Total extra-import data: the total import is identified as partner "Extra_Total"
extra_total <- datatrade_EU$extra_import %>% filter(partner=="Extra_Total")
# Extra-import data from countries where the pest is present
extra_pest <- datatrade_EU$extra_import %>% filter(partner!="Extra_Total")
# Intra-trade data
intra_trade  <- datatrade_EU$intra_trade
# Internal production data
internal_production  <- datatrade_EU$internal_production
# Sample 5 countries from data
filter_IDs <- sample(unique(extra_total$reporter), 5)
# Generate trade data (TradeData object)
trade_EU <- trade_data(extra_total = extra_total,
                       extra_pest = extra_pest,
                       intra_trade = intra_trade,
                       internal_production = internal_production,
                       filter_IDs = filter_IDs,
                       filter_period = 2020)
# Plot the total available quantity of commodity available in each country
plot_countries(data = trade_EU$total_trade,
               iso_col = "country_IDs",
               values_col = "total_available") +
  xlim(-30,50) + ylim(25,70)
```

Index