Package: ATQ (via r-universe)

September 17, 2024

Title Alert Time Quality - Evaluating Timely Epidemic Metrics

Version 0.2.3

Date 2024-07-28

Description Provides tools for evaluating timely epidemic detection models within school absenteeism-based surveillance systems. Introduces the concept of alert time quality as an evaluation metric. Includes functions to simulate populations, epidemics, and alert metrics associated with epidemic spread using population census data. The methods are based on research published in Vanderkruk et al. (2023) <doi:10.1186/s12889-023-15747-z> and Ward et al. (2019) <doi:10.1186/s12889-019-7521-7>.

License GPL (>= 3)

URL https://github.com/vjoshy/ATQ_Surveillance_Package

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, cli, fansi, farver, utf8, devtools, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr, purrr, zoo, ggplot2, gridExtra, rlang, scales

Depends R (>= 2.10)

LazyData true

VignetteBuilder knitr

BugReports https://github.com/vjoshy/ATQ_Surveillance_Package/issues

Repository https://vjoshy.r-universe.dev

RemoteUrl https://github.com/vjoshy/atq_surveillance_package

RemoteRef HEAD

RemoteSha a68e218842d0f6add062c6f46a3e6f3c312f8392

Contents

`alarm_metrics` *Create an Alarm Metrics Object*

Description

This function creates an object of class "alarm_metrics" from a list of metric matrices. It's used to organize and structure the results from various epidemic alarm metrics calculations.

Usage

```
alarm_metrics(metrics_list)
```

Arguments

`metrics_list`	A list containing matrices for different alarm metrics (FAR, ADD, AATQ, FATQ, WAATQ, WFATQ).

Value

An object of class "alarm_metrics" which is a list with the input metrics.

Examples

```
# Generate sample alarm metrics data
set.seed(123)
generate_metric_matrix <- function() {
  matrix(runif(15 * 11), nrow = 15, ncol = 11,
         dimnames = list(paste("Lag", 1:15),
                         paste("Threshold", seq(0.1, 0.6, by = 0.05))))
}

sample_metrics <- list(
  FAR = generate_metric_matrix(),
  ADD = generate_metric_matrix(),
  AATQ = generate_metric_matrix(),
  FATQ = generate_metric_matrix(),
  WAATQ = generate_metric_matrix(),
  WFATQ = generate_metric_matrix()
)

# Create an alarm_metrics object
alarm_metrics_obj <- alarm_metrics(sample_metrics)

# Check the class of the resulting object
class(alarm_metrics_obj)

# Access a specific metric
head(alarm_metrics_obj$AATQ)

# Use with other functions (assuming they're defined in your package)
# plot(alarm_metrics_obj, metric = "FAR")
```

---

alarm_plot_data *Create Alarm Plot Data Object*

Description

This function creates an object of class "alarm_plot_data" from epidemic data and best models. It ensures that the input data is in the correct format for plotting.

Usage

```
alarm_plot_data(epidemic_data, best_models)
```

Arguments

epidemic_data	A data frame or list of data frames containing epidemic data.
best_models	A list of data frames containing the best models for each metric.

Value

An object of class "alarm_plot_data" containing the epidemic data and best models.

Examples

```
# Generate sample epidemic data
set.seed(123)
epidemic_data <- data.frame(
  Date = rep(1:300, 3),
  ScYr = rep(1:3, each = 300),
  pct_absent = runif(900, 0, 0.1),
  lab_conf = rpois(900, lambda = 5),
  ref_date = rep(c(rep(0, 299), 1), 3),
  Case = rbinom(900, 1, 0.1),
  window = rep(c(rep(0, 280), rep(1, 20)), 3)
)

# Generate sample best models data
generate_best_model <- function() {
  data.frame(
    Date = sample(1:300, 50),
    ScYr = sample(1:3, 50, replace = TRUE),
    Alarm = sample(0:1, 50, replace = TRUE),
    lag = sample(1:15, 50, replace = TRUE),
    thres = runif(50, 0, 1)
  )
}

best_models <- list(
  AATQ = generate_best_model(),
  FATQ = generate_best_model(),
  FAR = generate_best_model(),
  ADD = generate_best_model(),
  WFATQ = generate_best_model(),
  WAATQ = generate_best_model()
)

# Create alarm_plot_data object
plot_data <- alarm_plot_data(epidemic_data, best_models)

# Check the structure of the resulting object
str(plot_data)
```

`best_model` *Create Best Model*

Description

This function sets the class of the given model data to "best_model".

Usage

```
best_model(model_data)
```

Arguments

model_data	A data frame containing the model data.

Value

An object of class "best_model" containing the model data.

Examples

```
# Generate sample model data
sample_model_data <- data.frame(
  ScYr = rep(1:3, each = 100),
  Date = rep(1:100, 3),
  Alarm = sample(c(0, 1), 300, replace = TRUE, prob = c(0.9, 0.1)),
  lag = sample(1:15, 300, replace = TRUE),
  thres = runif(300, 0.1, 0.6)
)

# Create best model
best_model_data <- best_model(sample_model_data)

# Print the class of the best model
class(best_model_data)
```

---

catchment_sim *Simulating catchment areas*

Description

Function to simulate specified catchments of square area (a x a). The number of schools in each catchment area is simulated via a specified distribution function, with gamma distribution as the default.

Usage

```
catchment_sim(n, area, dist_func = stats::rgamma, ...)
```

Arguments

n	number of catchments to be simulated
area	square dimension for catchment (if area = 20, then each catchment dimensions will be 20 x 20)
dist_func	distribution function to simulate number of schools, default is stats::rgamma
...	additional arguments passed to the distribution function

Value

A data frame with n rows and the following columns:

catchID	Unique identifier for each catchment area
num.schools	Number of schools in the catchment area
xStart	Starting x-coordinate of the catchment area
xEnd	Ending x-coordinate of the catchment area
yStart	Starting y-coordinate of the catchment area
yEnd	Ending y-coordinate of the catchment area

Examples

```
# Using default gamma distribution
catch_df1 <- catchment_sim(16, 20, shape = 4.1, rate = 2.7)

# Using normal distribution
catch_df2 <- catchment_sim(16, 20, dist_func = stats::rnorm, mean = 5, sd = 1)

# Using Poisson distribution
catch_df3 <- catchment_sim(16, 20, dist_func = stats::rpois, lambda = 3)
```

---

compile_epi *Compile Epidemic Data*

Description

Compiles and processes epidemic data, simulating school absenteeism using epidemic and individual data.

Usage

```
compile_epi(epidemic, individual_data, lags = 16, inf_period = 4, T = 300)
```

Arguments

epidemic	A list of simulated epidemic data (output from simepi() or similar function).
individual_data	A data frame of simulated individuals data (output from simulate_households() or similar function).
lags	Maximum number of lags for absenteeism, default = 16 (note that lag of zero is included).
inf_period	Infection period of epidemic, default = 4.
T	Time period, default = 300 days.

Value

A data frame containing compiled epidemic data, including:

`Date`	Day of the epidemic
`ScYr`	School year
`pct_absent`	Percentage of students absent
`absent`	Number of students absent
`absent_sick`	Number of students absent due to illness
`new_inf`	Number of new infections
`reported_cases`	Number of reported cases
`Case`	Indicator for lab confirmed flu case that day
`sinterm`, `costerm`	Seasonal terms
`window`	Indicator for "True Alarm" window
`ref_date`	Indicator for reference date
`lag0`, `lag1`, ..., `lag15`	Lagged absenteeism values

Examples

```
# Assuming you have previously simulated epidemic and individual data:
epidemic <- ssir(1000, alpha = 0.4)
individual_data <- data.frame(elem_child_ind = sample(0:1, 1000, replace = TRUE),
                              schoolID = sample(1:10, 1000, replace = TRUE))

compiled_data <- compile_epi(epidemic, individual_data)
```

---

create_alarm_metrics_summary

Create Alarm Metrics Summary

Description

This function creates a summary of alarm metrics, including statistics for each metric, best model parameters, reference dates, and best prediction dates for each epidemic year.

Usage

```
create_alarm_metrics_summary(metrics, best_models, epidemic_data)
```

Arguments

`metrics`	An object of class "alarm_metrics" containing matrices for different metrics.
`best_models`	A list of data frames, each containing the best model for a specific metric.
`epidemic_data`	A data frame containing the epidemic data, including ScYr, Date, and ref_date columns.

Value

An object of class "alarm_metrics_summary" containing summary statistics, best values, reference dates, and best prediction dates for each metric and epidemic year.

Examples

```
# Generate sample data
set.seed(123)

# Generate sample metrics
generate_metric_matrix <- function() {
  matrix(runif(15 * 11), nrow = 15, ncol = 11,
         dimnames = list(paste("Lag", 1:15),
                         paste("Threshold", seq(0.1, 0.6, by = 0.05))))
}

sample_metrics <- list(
  FAR = generate_metric_matrix(),
  ADD = generate_metric_matrix(),
  AATQ = generate_metric_matrix(),
  FATQ = generate_metric_matrix(),
  WAATQ = generate_metric_matrix(),
  WFATQ = generate_metric_matrix()
)

metrics <- structure(sample_metrics, class = c("alarm_metrics", "list"))

# Generate sample best models
generate_best_model <- function() {
  data.frame(
    ScYr = rep(1:3, each = 100),
    Date = rep(1:100, 3),
    Alarm = sample(c(0, 1), 300, replace = TRUE, prob = c(0.9, 0.1)),
    lag = sample(1:15, 300, replace = TRUE),
    thres = runif(300, 0.1, 0.6)
  )
}

best_models <- list(
  best.FAR = generate_best_model(),
  best.ADD = generate_best_model(),
  best.AATQ = generate_best_model(),
  best.FATQ = generate_best_model(),
  best.WAATQ = generate_best_model(),
  best.WFATQ = generate_best_model()
)

# Generate sample epidemic data
epidemic_data <- data.frame(
  ScYr = rep(1:3, each = 365),
  Date = rep(1:365, 3),
  ref_date = c(rep(0, 364), 1)
```

```
)

# Create alarm metrics summary
summary <- create_alarm_metrics_summary(metrics, best_models, epidemic_data)

# Print summary
print(summary)
```

`elementary_pop` *Create Elementary Schools population size*

Description

Function to simulate elementary school size and assigns it to catchments. The school population is simulated using a specified distribution function, with gamma distribution as the default.

Usage

```
elementary_pop(df, dist_func = stats::rgamma, ...)
```

Arguments

`df`	output data frame from catchment_sim function
`dist_func`	distribution function to simulate school population, default is stats::rgamma
`...`	additional arguments passed to the distribution function

Value

A data frame with the following columns:

`catchID`	Identifier for the catchment area
`schoolID`	Unique identifier for each school
`schoolPop`	Simulated population of the school
`xStart`	Starting x-coordinate of the catchment area
`xEnd`	Ending x-coordinate of the catchment area
`yStart`	Starting y-coordinate of the catchment area
`yEnd`	Ending y-coordinate of the catchment area

Examples

```
# Simulate catchment areas
catch_df <- catchment_sim(16, 20, shape = 3.5, rate = 2.8)

# Simulate elementary schools using default gamma distribution
elementary_df1 <- elementary_pop(catch_df, shape = 5.1, rate = 0.015)

# Simulate elementary schools using normal distribution
elementary_df2 <- elementary_pop(catch_df, dist_func = stats::rnorm,
                                 mean = 300, sd = 50)

# Simulate elementary schools using Poisson distribution
elementary_df3 <- elementary_pop(catch_df, dist_func = stats::rpois,
                                 lambda = 250)
```

`eval_metrics` *Evaluate Alarm Metrics for Epidemic Models*

Description

This function calculates various performance metrics for epidemic alarm systems across different lags and thresholds. It evaluates False Alarm Rate (FAR), Added Days Delayed (ADD), Average Alarm Time Quality (AATQ), First Alarm Time Quality (FATQ), and their weighted versions (WAATQ, WFATQ).

Usage

```
eval_metrics(data, maxlag = 15, thres = seq(0.1, 0.6, by = 0.05), topt = 14)
```

Arguments

`data`	A data frame containing the epidemic data with lagged variables, typically output from the compile_epi function.
`maxlag`	The maximum number of lags to consider (default: 15).
`thres`	A vector of threshold values to evaluate (default: seq(0.1, 0.6, by = 0.05)).
`topt`	Optimal alarm day (default = 14).

Value

A list containing three elements:

`metrics` An object of class "alarm_metrics" with the following components:

- FAR: Matrix of False Alarm Rates for each lag and threshold
- ADD: Matrix of Added Days Delayed for each lag and threshold
- AATQ: Matrix of Average Alarm Time Quality for each lag and threshold
- FATQ: Matrix of First Alarm Time Quality for each lag and threshold

- WAATQ: Matrix of Weighted Average Alarm Time Quality for each lag and threshold
- WFATQ: Matrix of Weighted First Alarm Time Quality for each lag and threshold
- best.AATQ: Best model according to AATQ
- best.FATQ: Best model according to FATQ
- best.FAR: Best model according to FAR
- best.ADD: Best model according to ADD
- best.WFATQ: Best model according to WFATQ
- best.WAATQ: Best model according to WAATQ

`plot_data` An object of class "alarm_plot_data" for generating plots

`summary` An object of class "alarm_metrics_summary" containing summary statistics

See Also

ssir, compile_epi

Examples

```
# Generate simulated epidemic data
n_rows <- 7421
n_houses <- 1000

epidemic_new <- ssir(n_rows, T = 300, alpha = 0.298, inf_init = 32, rep = 3)

individual_data <- data.frame(
  houseID = rep(1:n_houses, each = ceiling(n_rows / n_houses))[1:n_rows],
  catchID = sample(1:10, n_rows, replace = TRUE),
  schoolID = sample(1:10, n_rows, replace = TRUE),
  num_people = round(rnorm(n_rows, mean = 4, sd = 1)),
  num_elem_child = round(rnorm(n_rows, mean = 1, sd = 1)),
  xStart = 0,
  xEnd = 5,
  yStart = 0,
  yEnd = 5,
  loc.x = rnorm(n_rows, mean = 2.5, sd = 1),
  loc.y = rnorm(n_rows, mean = 2.5, sd = 1),
  individualID = 1:n_rows,
  elem_child_ind = sample(0:1, n_rows, replace = TRUE)
)

compiled_data <- compile_epi(epidemic_new, individual_data)

# Evaluate alarm metrics
alarm_metrics <- eval_metrics(compiled_data,
                              thres = seq(0.1, 0.3, by = 0.05))

# Access the results
summary(alarm_metrics$summary)
```

`plot.alarm_metrics` *Plot Heatmap of Alarm Metrics*

Description

This function creates a heatmap visualization of the specified alarm metric across different lags and thresholds.

Usage

```
## S3 method for class 'alarm_metrics'
plot(x, metric = "AATQ", col = heat.colors(12), ...)
```

Arguments

`x`	An object of class "alarm_metrics" containing matrices of metric values.
`metric`	A character string specifying which metric to plot. Default is "AATQ". Options include "FAR", "ADD", "AATQ", "FATQ", "WAATQ", "WFATQ".
`col`	Set heat map color profile, default is heat.colors(15)
`...`	Additional arguments passed to the image function.

Value

A heatmap plot of the specified metric.

Examples

```
# Generate sample alarm metrics data
set.seed(123)
generate_metric_matrix <- function() {
  matrix(runif(15 * 11), nrow = 15, ncol = 11,
         dimnames = list(paste("Lag", 1:15),
                         paste("Threshold", seq(0.1, 0.6, by = 0.05))))
}

sample_metrics <- list(
  FAR = generate_metric_matrix(),
  ADD = generate_metric_matrix(),
  AATQ = generate_metric_matrix(),
  FATQ = generate_metric_matrix(),
  WAATQ = generate_metric_matrix(),
  WFATQ = generate_metric_matrix(),
 lags = 1:15,
 thresholds = seq(0.1, 0.6, by = 0.05)
)

# Create an alarm_metrics object
alarm_metrics_obj <- structure(sample_metrics, class = c("alarm_metrics", "list"))
```

```
# Plot the heatmap for AATQ (default)
plot(alarm_metrics_obj)

# Plot the heatmap for FAR
plot(alarm_metrics_obj, metric = "FAR")

# Customize the plot
plot(alarm_metrics_obj, metric = "FATQ", col = heat.colors(12))
```

`plot.alarm_plot_data` *Plot Epidemic Data with Alarm Metrics*

Description

This function creates a series of plots, one for each epidemic year, showing the absenteeism percentage, laboratory confirmed cases, reference dates, and alarm points for different models.

Usage

```
## S3 method for class 'alarm_plot_data'
plot(x, ...)
```

Arguments

`x`	An object of class "alarm_plot_data" containing epidemic data and best models.
`...`	Additional arguments passed to the plotting function (currently unused).

Value

A list of ggplot objects, one for each epidemic year.

Examples

```
# Generate sample data
# Generate simulated epidemic data
n_rows <- 7421
n_houses <- 1000

epidemic_new <- ssir(n_rows, T = 300, alpha =0.298, inf_init = 32, rep = 3)

 individual_data <- data.frame(
houseID = rep(1:n_houses, each = ceiling(n_rows / n_houses))[1:n_rows],
catchID = sample(1:10, n_rows, replace = TRUE),
schoolID = sample(1:10, n_rows, replace = TRUE),
num_people = round(rnorm(n_rows, mean = 4, sd = 1)),  # Normal distribution for num_people
num_elem_child = round(rnorm(n_rows, mean = 1, sd = 1)),  # Normal distribution for num_elem_child
xStart = 0,
```

```
xEnd = 5,
yStart = 0,
yEnd = 5,
loc.x = rnorm(n_rows, mean = 2.5, sd = 1),  # Normal distribution for loc.x
loc.y = rnorm(n_rows, mean = 2.5, sd = 1),  # Normal distribution for loc.y
individualID = 1:n_rows,
elem_child_ind = sample(0:1, n_rows, replace = TRUE)
)

compiled_data <- compile_epi(epidemic_new, individual_data)

# Evaluate alarm metrics
results <- eval_metrics(compiled_data, thres = seq(0.1,0.6,by = 0.05))

# Generate plots
plots <- plot(results$plot_data)

# Display the first plot
if(interactive()) print(plots[[1]])
```

`plot.ssir_epidemic` *Plot method for a single SSIR epidemic simulation*

Description

This function creates plots for a single SSIR (Stochastic Susceptible-Infectious-Removed) epidemic simulation. It generates two plots: one for new infections and one for reported cases.

Usage

```
## S3 method for class 'ssir_epidemic'
plot(x, ...)
```

Arguments

`x`	An object of class "ssir_epidemic", typically the result of calling ssir() with rep = NULL or 1.
`...`	Additional arguments passed to the underlying plotting function (currently unused).

Value

A grid arrangement of two ggplot objects: new infections and reported cases.

See Also

`plot_single_epidemic` for the underlying plotting function

Examples

```
# Run a single simulation
result <- ssir(N = 10000, T = 300, alpha = 0.3, inf_period = 4, inf_init = 32,
report = 0.02, lag = 7)

# Plot the results
plot(result)
```

`plot.ssir_epidemic_multi` *Plot method for multiple SSIR epidemic simulations*

Description

This function creates plots for multiple SSIR (Stochastic Susceptible-Infectious-Removed) epidemic simulations. It generates two plots (new infections and reported cases) for each simulation in the multi-simulation object.

Usage

```
## S3 method for class 'ssir_epidemic_multi'
plot(x, ...)
```

Arguments

`x`	An object of class "ssir_epidemic_multi", typically the result of calling ssir() with rep > 1.
`...`	Additional arguments passed to the underlying plotting function (currently unused).

Value

A list of grid arrangements, each containing two ggplot objects (new infections and reported cases) for each epidemic simulation.

See Also

`plot_single_epidemic` for the underlying plotting function

Examples

```
# Run multiple simulations
multi_result <- ssir(N = 10000, T = 300, alpha = 0.3, inf_period = 4,
                     inf_init = 32, report = 0.02, lag = 7, rep = 5)

# Plot the results
plots <- plot(multi_result)
```

```
# Display the first simulation's plots
plots[[1]]
```

`plot_single_epidemic` *Plot a single epidemic simulation*

Description

This function creates two plots for a single epidemic simulation: one for new infections and one for reported cases.

Usage

```
plot_single_epidemic(epidemic, epidemic_num)
```

Arguments

`epidemic`	A list containing the results of a single epidemic simulation, typically an element from an "ssir_epidemic_multi" object.
`epidemic_num`	An integer indicating the epidemic number, used in the plot titles.

Value

A grid arrangement of two ggplot objects: new infections and reported cases.

Examples

```
# Assuming you have run a simulation:
result <- ssir(N = 10000, T = 300, alpha = 0.3, inf_period = 4, inf_init = 32,
report = 0.02, lag = 7)
plot_single_epidemic(result, 1)

# For multiple simulations:
multi_result <- ssir(N = 10000, T = 300, alpha = 0.3, inf_period = 4, inf_init = 32,
report = 0.02, lag = 7, rep = 5)
plot_single_epidemic(multi_result[[1]], 1)
```

simulate_households *Simulate total households and individuals data*

Description

Creates two simulated populations, households and individuals

Usage

```
simulate_households(children_df, noChildren_df)
```

Arguments

children_df	data frame output from house_children() function
noChildren_df	data frame output from house_noChildren() function

Value

list of two data frames; simulated individuals data and simulated households data

Examples

```
# Simulate catchment areas
catch_df <- catchment_sim(4, 5, shape = 2.5, rate = 1.8)

# Simulate elementary schools using default gamma distribution
elementary_df <- elementary_pop(catch_df, shape = 4.1, rate = 0.019)

# Simulate households with children
house_children <- subpop_children(elementary_df, n = 2,
                                  prop_parent_couple = 0.7,
                                  prop_children_couple = c(0.3, 0.5, 0.2),
                                  prop_children_lone = c(0.4, 0.4, 0.2),
                                  prop_elem_age = 0.2)

# Simulate households without children using pre-specified proportions
house_nochildren <- subpop_noChildren(house_children, elementary_df,
                                   prop_house_size = c(0.2, 0.3, 0.25, 0.15, 0.1),
                                   prop_house_Children = 0.3)

# simulate households and individuals data
simulation <- simulate_households(house_children, house_nochildren)
```

`ssir` *Stochastic SIR simulation*

Description

This function runs a stochastic Susceptible-Infectious-Removed (SIR) epidemic simulation. It can run a single simulation or multiple replications. Also simulates laboratory confirmed cases

Usage

```
ssir(
  N,
  T = 300,
  alpha,
  avg_start = 45,
  min_start = 20,
  inf_period = 4,
  inf_init = 32,
  report = 0.02,
  lag = 7,
  rep = NULL
)
```

Arguments

`N`	Numeric. The total population size.
`T`	Numeric. The duration of the simulation in time steps. Default is 300.
`alpha`	Numeric. The transmission rate. Must be a number between 0 and 1.
`avg_start`	Numeric. The average start day of the epidemic. Default is 45.
`min_start`	Numeric. The minimum start day of the epidemic. Default is 20.
`inf_period`	Numeric. The duration of the infectious period in time steps. Default is 4.
`inf_init`	Numeric. The initial number of infected individuals. Default is 32.
`report`	Numeric. The proportion of cases that are reported. Default is 0.02.
`lag`	Numeric. The average delay in reporting cases. Default is 7.
`rep`	Numeric or NULL. The number of simulation replications to run. If NULL or 1, a single simulation is run.

Value

If rep is NULL or 1, returns an object of class "ssir_epidemic" containing:

`new_inf`	Vector of new infections at each time step
`reported_cases`	Vector of reported cases at each time step
`S`	Vector of susceptible individuals at each time step

I	Vector of infectious individuals at each time step
R	Vector of removed individuals at each time step
parameters	List of input parameters

If rep > 1, returns an object of class "ssir_epidemic_multi" containing multiple simulation results.

Examples

```
# Run a single simulation
result <- ssir(N = 10000, T = 300, alpha = 0.3, inf_period = 4,
inf_init = 32, report = 0.02, lag = 7)

# Run multiple simulations
multi_result <- ssir(N = 10000, T = 300, alpha = 0.3, inf_period = 4, inf_init = 32,
report = 0.02, lag = 7, rep = 100)
```

---

subpop_children *Simulate households with children*

Description

Simulation of households with children using specified random distributions. Number of observations is multiplied by a default value of five.

Usage

```
subpop_children(
  df,
  n = 5,
  prop_parent_couple = NULL,
  prop_children_couple = NULL,
  prop_children_lone = NULL,
  prop_elem_age = NULL,
  parent_dist = stats::runif,
  child_dist = stats::runif,
  age_dist = stats::runif,
  ...
)
```

Arguments

df	simulated output data frame from elementary_pop function
n	population multiplier, default value = 5
prop_parent_couple	proportion of parents as a couple (optional)

prop_children_couple	vector of proportions for coupled parents with 1, 2, 3+ children (optional)
prop_children_lone	vector of proportions for single parents with 1, 2, 3+ children (optional)
prop_elem_age	proportion of children that are of elementary school age (optional)
parent_dist	distribution function for parent type, default is stats::runif
child_dist	distribution function for number of children, default is stats::runif
age_dist	distribution function for child age, default is stats::runif
...	additional arguments passed to the distribution functions

Details

This function can be used interactively or with pre-specified parameters. If proportions are not provided, the user will be prompted to enter them. Custom distribution functions can be specified for parent type, number of children, and child age. The total number of simulations (n * sum of school populations) is automatically passed as the first argument to each distribution function.

Value

A data frame representing the simulated population of households with children, including:

schoolID	Assigned school ID for the household
houseID	Unique identifier for each household
num_parent	Number of parents in the household (1 or 2)
num_child	Total number of children in the household
num_elem_child	Number of elementary school-aged children in the household
catchID	Assigned catchment ID for the household
schoolPop	Total population of elementary school assigned for the household
xStart	Starting X-coordindate for assigned catchment
xEnd	End X-coordindate for assigned catchment
yStart	Starting Y-coordindate for assigned catchment
yEnd	End Y-coordindate for assigned catchment
schoolID	Assigned school ID for the household
num_people	Total number of people in the household

Examples

```
# Simulate catchment area
catch_df <- catchment_sim(4, 5, shape = 2.5, rate = 1.3)

# Simulate elementary schools using default gamma distribution
elementary_df <- elementary_pop(catch_df, shape = 3.3, rate = 0.015)

# Simulate households with children
house_children <- subpop_children(elementary_df, n = 2,
```

```
                                  prop_parent_couple = 0.7,
                                  prop_children_couple = c(0.3, 0.5, 0.2),
                                  prop_children_lone = c(0.4, 0.4, 0.2),
                                  prop_elem_age = 0.6)

# Using custom distributions
house_children2 <- subpop_children(elementary_df, n = 3,
                                  prop_parent_couple = 0.7,
                                  prop_children_couple = c(0.3, 0.5, 0.2),
                                  prop_children_lone = c(0.4, 0.4, 0.2),
                                  prop_elem_age = 0.6,
                                  parent_dist = stats::rnorm, mean = 0.5, sd = 0.1,
                                  child_dist = stats::rbeta, shape1 = 2, shape2 = 2,
                                  age_dist = stats::runif)
```

`subpop_noChildren` *Simulate households without children*

Description

Simulation of households without children using data frames from elementary_pop() and subpop_children() functions.

Usage

```
subpop_noChildren(
  df,
  df2,
  prop_house_size = NULL,
  prop_house_Children = NULL,
  house_size = NULL
)
```

Arguments

`df`	simulated output data frame from subpop_children function
`df2`	simulated output data frame from elementary_pop function
`prop_house_size`	vector of proportions for households with 1, 2, 3, 4, 5+ members (optional)
`prop_house_Children`	proportion of households with children (optional)
`house_size`	vector of random numbers for household size simulation (optional)

Details

This function can be used interactively or with pre-specified parameters. If proportions are not provided, the user will be prompted to enter them. The function calculates the number of households without children for each catchment area based on the proportion of households with children.

Value

A data frame representing the simulated population of households without children, including:

catchID	Catchment area ID
houseID	Unique identifier for each household
num_people	Number of people in the household
schoolPop	Total population of elementary school assigned for the household
xStart	Starting X-coordindate for assigned catchment
xEnd	End X-coordindate for assigned catchment
yStart	Starting Y-coordindate for assigned catchment
yEnd	End Y-coordindate for assigned catchment

Examples

```
# Simulate catchment areas
catch_df <- catchment_sim(4, 5, shape = 2.5, rate = 1.8)

# Simulate elementary schools using default gamma distribution
elementary_df <- elementary_pop(catch_df, shape = 3.1, rate = 0.015)

# Simulate households with children
house_children <- subpop_children(elementary_df, n = 3,
                                  prop_parent_couple = 0.7,
                                  prop_children_couple = c(0.3, 0.5, 0.2),
                                  prop_children_lone = c(0.4, 0.4, 0.2),
                                  prop_elem_age = 0.6)

# Simulate households without children using pre-specified proportions
house_noChild <- subpop_noChildren(house_children, elementary_df,
                                   prop_house_size = c(0.2, 0.3, 0.25, 0.15, 0.1),
                                   prop_house_Children = 0.3)
```

---

summary.alarm_metrics_summary

Summary of Alarm Metrics

Description

This function provides a summary of the alarm metrics, including mean and variance for each metric, the best lag and threshold values, and lists of reference dates and best prediction dates.

Usage

```
## S3 method for class 'alarm_metrics_summary'
summary(object, ...)
```

Arguments

object	A list of class alarm_metrics_summary containing the alarm metrics summary data.
...	Additional arguments passed to the function (currently unused).

Value

A printed summary of the alarm metrics.

Examples

```
set.seed(123)

# Generate sample metrics
generate_metric_matrix <- function() {
  matrix(runif(15 * 11), nrow = 15, ncol = 11,
         dimnames = list(paste("Lag", 1:15),
                         paste("Threshold", seq(0.1, 0.6, by = 0.05))))
}

sample_metrics <- list(
  FAR = generate_metric_matrix(),
  ADD = generate_metric_matrix(),
  AATQ = generate_metric_matrix(),
  FATQ = generate_metric_matrix(),
  WAATQ = generate_metric_matrix(),
  WFATQ = generate_metric_matrix()
)

metrics <- structure(sample_metrics, class = c("alarm_metrics", "list"))

# Generate sample best models
generate_best_model <- function() {
  data.frame(
    ScYr = rep(1:3, each = 100),
    Date = rep(1:100, 3),
    Alarm = sample(c(0, 1), 300, replace = TRUE, prob = c(0.9, 0.1)),
    lag = sample(1:15, 300, replace = TRUE),
    thres = runif(300, 0.1, 0.6)
  )
}

best_models <- list(
  best.FAR = generate_best_model(),
  best.ADD = generate_best_model(),
  best.AATQ = generate_best_model(),
  best.FATQ = generate_best_model(),
  best.WAATQ = generate_best_model(),
  best.WFATQ = generate_best_model()
)
```

```
# Generate sample epidemic data
epidemic_data <- data.frame(
  ScYr = rep(1:3, each = 365),
  Date = rep(1:365, 3),
  ref_date = c(rep(0, 364), 1)
)

# Create alarm metrics summary
data <- create_alarm_metrics_summary(metrics, best_models, epidemic_data)

# Print summary
summary(data)
```

`summary.ssir_epidemic` *S3 Summary method for epidemic Simulation*

Description

This function provides a summary of SSIR epidemic simulation. It calculates and displays the statistics.

Usage

```
## S3 method for class 'ssir_epidemic'
summary(object, ...)
```

Arguments

`object`	An object of class "ssir_epidemic", typically the result of calling ssir() with rep = 1 or NULL.
`...`	Additional arguments affecting the summary produced.

Value

No return value, called for side effects.

Examples

```
# Run multiple simulations
result <- ssir(N = 10000, T = 300, alpha = 0.3, inf_period = 4,
                    inf_init = 32, report = 0.02, lag = 7)

# Display summary
summary(result)
```

summary.ssir_epidemic_multi *S3 Summary method for multiple epidemic simulations*

Description

This function provides a summary of multiple SSIR epidemic simulations. It calculates and displays average statistics across all simulations.

Usage

```
## S3 method for class 'ssir_epidemic_multi'
summary(object, ...)
```

Arguments

object	An object of class "ssir_epidemic_multi", typically the result of calling ssir() with rep > 1.
...	Additional arguments affecting the summary produced.

Value

No return value, called for side effects.

Examples

```
# Run multiple simulations
multi_result <- ssir(N = 10000, T = 300, alpha = 0.3, inf_period = 4,
                     inf_init = 32, report = 0.02, lag = 7, rep = 100)

# Display summary
summary(multi_result)
```

Index