Package: GeoEpiAI (via r-universe)

November 18, 2026

Type Package

Title GeoAI and Digital Epidemiology Tools for Disease Surveillance

Version 0.1.0

Author Vinodhkumar Obli Rajendran [aut, cre], Keerthi Aaradhana [aut]

Maintainer Vinodhkumar Obli Rajendran

<vinodhkumar.rajendran@gmail.com>

Description Provides tools for digital epidemiology and geospatial artificial intelligence, including surveillance-data cleaning, spatial epidemiology, spatial autocorrelation and hotspot detection, environmental data integration, machine-learning-based disease-risk prediction, dynamic risk mapping, early-warning detection, and automated epidemiological reporting. The methods include spatial autocorrelation (Moran, 1950) <doi:10.1093/biomet/37.1-2.17>, local spatial association analysis (Getis and Ord, 1992) <doi:10.1111/j.1538-4632.1992.tb00261.x>, Random Forest (Breiman, 2001) <doi:10.1023/A:1010933404324>, XGBoost (Chen and Guestrin, 2016) <doi:10.1145/2939672.2939785>, and spatial data handling (Pebesma, 2018) <doi:10.32614/RJ-2018-009>. The package supports applications in disease surveillance, epidemiological risk assessment, and environmental health.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, lubridate, randomForest, sf, spdep, xgboost

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-11-17 12:20:11 UTC

RemoteUrl https://github.com/cran/GeoEpiAI

RemoteRef HEAD

RemoteSha 72284699726aedfe53ec22488a2753492b9c7243

Contents

clean_surveillance 2
dynamic_risk_map 3
early_warning 4
epi_report 6
geoepi_pipeline 7
hotspot_detection 8
integrate_environment 9
predict_risk 10
spatial_autocorrelation 11
train_rf_risk 12
train_xgb_risk 13
validate_surveillance 14

Index **16**

`clean_surveillance` *Validate and clean disease surveillance data*

Description

Cleans disease surveillance records by standardising location, date, and case-count variables. Optionally validates population data, aggregates duplicate location-date records, and calculates monthly incidence per 100,000 population.

Usage

```
clean_surveillance(
  data,
  location,
  date,
  cases,
  population = NULL,
  aggregate = TRUE
)
```

Arguments

data	A data.frame containing surveillance records.
location	Column containing geographic unit.
date	Column containing date/month.
cases	Column containing disease cases.
population	Optional population column.
aggregate	If TRUE, aggregate duplicate location-date records.

Value

A cleaned data.frame containing location, date, cases, month, and incidence. If population is supplied, the population column is also returned.

Examples

```
dat <- data.frame(
  location = c("A", "A", "B", "B"),
  date = as.Date(c(
    "2025-01-01",
    "2025-02-01",
    "2025-01-01",
    "2025-02-01"
  )),
  cases = c(10, 15, 5, 8),
  population = c(10000, 10000, 8000, 8000)
)

clean_surveillance(
  dat,
  location = "location",
  date = "date",
  cases = "cases",
  population = "population"
)
```

---

dynamic_risk_map *Create a dynamic risk map*

Description

Creates a ggplot-based spatial risk map with separate facets for each time period.

Usage

```
dynamic_risk_map(data, risk = "risk", time = "date")
```

Arguments

data	sf object containing predicted risk and time.
risk	Column containing predicted risk.
time	Column containing time period.

Value

A ggplot object faceted by time.

Examples

```
map_data <- sf::st_as_sf(
  data.frame(
    risk = c(0.2, 0.5, 0.7, 0.9),
    date = as.Date(c(
      "2025-01-01",
      "2025-01-01",
      "2025-02-01",
      "2025-02-01"
    )),
    x = c(0, 1, 0, 1),
    y = c(0, 0, 1, 1)
  ),
  coords = c("x", "y"),
  crs = 4326
)

dynamic_risk_map(
  map_data,
  risk = "risk",
  time = "date"
)
```

---

early_warning *Detect Early-Warning Signals in Surveillance Data*

Description

Calculates rolling baselines and Z-scores to identify potential increases in disease cases. When a location column is available, calculations are performed independently for each location. At least two previous observations are required to calculate a Z-score.

Usage

```
early_warning(data, date, cases, window = 4, threshold = 2, location = NULL)
```

Arguments

data	A data frame containing surveillance observations.
date	Character string specifying the date column.
cases	Character string specifying the case-count column.
window	Number of previous observations used for the rolling baseline. Must be at least 2.
threshold	Positive Z-score threshold for an alert.
location	Optional character string specifying the location column. If NULL, a column named "location" is detected automatically when available.

Details

The rolling baseline is calculated using previous observations only. Calculations are performed independently for each location. At least two previous observations are required to calculate the sample standard deviation and Z-score.

When the historical standard deviation is zero, an increase above the baseline is assigned an infinite Z-score and flagged as an alert. Otherwise, the Z-score is set to zero.

Value

A data frame containing date, cases, baseline, z_score, alert, and alert_level. A location column is retained when location-wise analysis is performed.

Examples

```
surveillance <- data.frame(
  location = rep(c("A", "B"), each = 8),
  date = rep(as.Date("2026-01-01") + 0:7, 2),
  cases = c(
    10, 11, 10, 12, 11, 13, 30, 35,
    2, 2, 3, 2, 3, 2, 3, 3
  )
)

early_warning(
  surveillance,
  date = "date",
  cases = "cases",
  window = 4,
  threshold = 2
)
```

`epi_report` *Generate an automated epidemiological report*

Description

Generates an HTML epidemiological report containing a surveillance summary, disease trend, optional risk map, and optional early-warning results.

Usage

```
epi_report(
  surveillance,
  output_file,
  title = "GeoEpiAI Epidemiological Report",
  risk_map = NULL,
  warning_table = NULL
)
```

Arguments

`surveillance`	Clean surveillance data containing at least `location`, `date`, and `cases` columns.
`output_file`	Explicit path and filename for the generated HTML report. No default output path is used.
`title`	Report title.
`risk_map`	Optional ggplot risk map.
`warning_table`	Optional early-warning table.

Value

Path to the generated report.

Examples

```
if (requireNamespace("rmarkdown", quietly = TRUE)) {
  surveillance <- data.frame(
    location = c("A", "A", "B", "B"),
    date = as.Date(c(
      "2025-01-01",
      "2025-02-01",
      "2025-01-01",
      "2025-02-01"
    )),
    cases = c(10, 15, 5, 8)
  )

  output_file <- file.path(
    tempdir(),
```

```
    "GeoEpiAI_example_report.html"
  )

  epi_report(
    surveillance = surveillance,
    output_file = output_file,
    title = "Example GeoEpiAI Report"
  )
}
```

`geoepi_pipeline` *Run the complete GeoEpiAI surveillance workflow*

Description

Runs the main GeoEpiAI surveillance workflow, including surveillance-data cleaning and early-warning analysis, with optional machine-learning risk prediction and spatial risk mapping.

Usage

```
geoepi_pipeline(
  surveillance,
  location,
  date,
  cases,
  population = NULL,
  risk_formula = NULL,
  sf_data = NULL
)
```

Arguments

`surveillance`	Surveillance data.frame.
`location`	Location column.
`date`	Date column.
`cases`	Case-count column.
`population`	Optional population column.
`risk_formula`	Optional binary risk formula.
`sf_data`	Optional sf spatial data for mapping.

Value

A named list of cleaned data, warning results and optional model and mapping results.

Examples

```
dat <- data.frame(
  location = c("A", "A", "B", "B"),
  date = as.Date(c(
    "2025-01-01",
    "2025-02-01",
    "2025-01-01",
    "2025-02-01"
  )),
  cases = c(10, 15, 5, 8),
  population = c(10000, 10000, 8000, 8000)
)

result <- geoepi_pipeline(
  surveillance = dat,
  location = "location",
  date = "date",
  cases = "cases",
  population = "population"
)

names(result)
```

---

hotspot_detection *Local Getis-Ord Gi* hotspot detection*

Description

Identifies statistically high and low spatial concentrations using the Local Getis-Ord Gi* statistic.

Usage

```
hotspot_detection(data, value, queen = TRUE)
```

Arguments

data	sf object.
value	Variable to analyse.
queen	Use queen contiguity for polygons.

Value

An sf object containing the Gi statistic, p-value, and hotspot classification.

Examples

```
pts <- sf::st_as_sf(
  data.frame(
    value = c(
      10, 20, 30, 40,
      20, 30, 40, 50,
      30, 40, 50, 60,
      40, 50, 60, 70
    ),
    x = rep(0:3, each = 4),
    y = rep(0:3, times = 4)
  ),
  coords = c("x", "y"),
  crs = 4326
)

hotspot_result <- hotspot_detection(
  pts,
  value = "value"
)

hotspot_result
```

---

integrate_environment *Integrate environmental predictors with surveillance locations*

Description

Supports tabular location/date joins and sf spatial joins.

Usage

```
integrate_environment(
  surveillance,
  environment,
  by = c("location", "date"),
  method = c("left", "inner", "spatial_nearest"),
  max_distance = Inf
)
```

Arguments

surveillance	Surveillance data.frame.
environment	Environmental data.frame or sf object.
by	Character vector of shared keys, e.g. c("location", "date").
method	Join method: "left", "inner", or "spatial_nearest".
max_distance	Maximum distance for nearest spatial join, in CRS units.

Value

Joined data.frame or sf object.

Examples

```
surveillance <- data.frame(
  location = c("A", "B"),
  date = as.Date(c("2025-01-01", "2025-01-01")),
  cases = c(10, 5)
)

environment <- data.frame(
  location = c("A", "B"),
  date = as.Date(c("2025-01-01", "2025-01-01")),
  precipitation = c(120, 80),
  temperature = c(25, 23)
)

integrate_environment(
  surveillance,
  environment,
  by = c("location", "date"),
  method = "left"
)
```

---

predict_risk *Predict disease risk from a GeoEpiAI model*

Description

Generates predicted disease-risk probabilities from a Random Forest or GeoEpiAI XGBoost model.

Usage

```
predict_risk(model, newdata)
```

Arguments

model	Random Forest, xgboost, or GeoEpiAI_xgb model.
newdata	New observations containing predictor variables.

Value

Numeric predicted risk probabilities.

Examples

```
set.seed(123)

dat <- data.frame(
  risk = factor(
    rep(c("Low", "High"), each = 10)
  ),
  precipitation = runif(20, 50, 200),
  temperature = runif(20, 20, 35)
)

rf <- train_rf_risk(
  risk ~ precipitation + temperature,
  data = dat,
  ntree = 20
)

new_data <- data.frame(
  precipitation = c(100, 150),
  temperature = c(25, 30)
)

predict_risk(
  rf,
  new_data
)
```

---

spatial_autocorrelation

Global Moran's I spatial autocorrelation

Description

Calculates Global Moran's I to assess spatial autocorrelation in an sf point or polygon dataset.

Usage

```
spatial_autocorrelation(data, value, queen = TRUE)
```

Arguments

data	sf polygon/point object.
value	Variable to analyse.
queen	Use queen contiguity for polygons.

Value

A list containing the Moran test and spatial weights.

Examples

```
pts <- sf::st_as_sf(
  data.frame(
    value = c(
      10, 20, 30, 40,
      20, 30, 40, 50,
      30, 40, 50, 60,
      40, 50, 60, 70
    ),
    x = rep(0:3, each = 4),
    y = rep(0:3, times = 4)
  ),
  coords = c("x", "y"),
  crs = 4326
)

moran_result <- spatial_autocorrelation(
  pts,
  value = "value"
)

moran_result$test
```

`train_rf_risk` *Train a Random Forest disease-risk model*

Description

Trains a Random Forest classifier for binary disease-risk prediction.

Usage

```
train_rf_risk(formula, data, ntree = 500, seed = 123)
```

Arguments

`formula`	Model formula; outcome should be binary 0/1 for classification.
`data`	Training data.
`ntree`	Number of trees.
`seed`	Random seed.

Value

A randomForest model.

Examples

```
set.seed(123)

dat <- data.frame(
  risk = factor(
    rep(c("Low", "High"), each = 10)
  ),
  precipitation = runif(20, 50, 200),
  temperature = runif(20, 20, 35)
)

rf <- train_rf_risk(
  risk ~ precipitation + temperature,
  data = dat,
  ntree = 20
)

rf
```

`train_xgb_risk` *Train an XGBoost disease-risk model*

Description

Trains an XGBoost binary classification model for disease-risk prediction.

Usage

```
train_xgb_risk(
  formula,
  data,
  nrounds = 200,
  max_depth = 4,
  eta = 0.05,
  seed = 123
)
```

Arguments

`formula`	Binary classification formula.
`data`	Training data.
`nrounds`	Number of boosting rounds.
`max_depth`	Maximum tree depth.
`eta`	Learning rate.
`seed`	Random seed.

Value

A list containing the xgboost model and preprocessing information.

Examples

```
set.seed(123)

dat <- data.frame(
  risk = rep(c(0, 1), each = 10),
  precipitation = runif(20, 50, 200),
  temperature = runif(20, 20, 35)
)

xgb <- train_xgb_risk(
  risk ~ precipitation + temperature,
  data = dat,
  nrounds = 10
)

xgb
```

---

validate_surveillance *Validate surveillance data without modifying it*

Description

Checks surveillance data for missing locations, missing dates, non-numeric case counts, and negative case counts.

Usage

```
validate_surveillance(data, location, date, cases)
```

Arguments

data	A data.frame.
location	Location column.
date	Date column.
cases	Case-count column.

Value

Invisibly returns TRUE.

Examples

```
dat <- data.frame(
  location = c("A", "B"),
  date = as.Date(c(
    "2025-01-01",
    "2025-02-01"
  )),
  cases = c(10, 5)
)

validate_surveillance(
  dat,
  location = "location",
  date = "date",
  cases = "cases"
)
```

Index

clean_surveillance, 2

dynamic_risk_map, 3

early_warning, 4
epi_report, 6

geoepi_pipeline, 7

hotspot_detection, 8

integrate_environment, 9

predict_risk, 10

spatial_autocorrelation, 11

train_rf_risk, 12
train_xgb_risk, 13

validate_surveillance, 14