Package ‘epitraxr’

September 14, 2025

Title Manipulate 'EpiTrax' Data and Generate Reports

Version 0.5.0

Description A fast, flexible tool for generating disease surveillance reports from data exported from 'EpiTrax', a central repository for epidemiological data used by public health officials. It provides functions to manipulate 'EpiTrax' datasets, tailor reports to internal or public use, and export reports in CSV, Excel 'xlsx', or PDF formats.

License MIT + file LICENSE

URL `https://epiforesite.github.io/epitraxr/`, `https://github.com/EpiForeSITE/epitraxr`

BugReports `https://github.com/EpiForeSITE/epitraxr/issues`

Depends R (>= 4.1.0)

Imports lubridate, stats, utils, writexl, yaml

Suggests DT, kableExtra, knitr, readxl, rmarkdown, shiny, shinyjs, stringr, tinytest

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Andrew Pulsipher [aut, cre] (ORCID: <`https://orcid.org/0000-0002-0773-3210`>), Nate Lanza [aut, ctb], Centers for Disease Control and Prevention's Center for Forecasting and Outbreak Analytics [fnd] (Cooperative agreement CDC-RFA-FT-23-0069)

Maintainer Andrew Pulsipher <`pulsipher.a@gmail.com`>

Repository CRAN

Date/Publication 2025-09-14 16:20:02 UTC

Contents

`clear_old_reports` *Clear out old reports before generating new ones.*

Description

`clear_old_reports` deletes reports from previous runs and returns a list of the reports that were deleted.

Usage

```
clear_old_reports(internal, public)
```

Arguments

`internal`	Filepath. Folder for internal reports.
`public`	Filepath. Folder for public reports.

Value

The list of old reports that were cleared.

Examples

```
ireports_folder <- file.path(tempdir(), "internal")
preports_folder <- file.path(tempdir(), "public")
dir.create(ireports_folder)
dir.create(preports_folder)

clear_old_reports(ireports_folder, preports_folder)
unlink(c(ireports_folder, preports_folder), recursive = TRUE)
```

compute_trend *Compute the report trend*

Description

'compute_trend' compares values of two columns and produces a new column containing the trend result. The trend is represented by one of three values:

- "Elevated": increase from baseline
- "Less Than Expected": decrease from baseline
- "Expected": no change from baseline

Usage

```
compute_trend(current, historical, threshold = 0)
```

Arguments

`current`	List. Current data.
`historical`	List. Historical comparison data.
`threshold`	Numeric. Percentage threshold (as decimal) for determining trend significance. Values within this percentage of the historical value are considered "Expected". Defaults to 0.0 (any difference triggers trend).

Value

Character vector containing the trend labels.

Examples

```
# Without threshold - any difference triggers trend
compute_trend(c(5, 10, 10), c(3, 10, 11))

# With 15% threshold - small changes are "Expected"
compute_trend(c(5, 10, 10), c(3, 10, 11), threshold = 0.15)
```

convert_counts_to_rate

Convert case counts to rate

Description

'convert_counts_to_rate' converts case counts for a given population to an adjusted per population of size X and rounds to the given number of digits.

Usage

```
convert_counts_to_rate(counts, pop, digits, rate_adj_pop = 1e+05)
```

Arguments

`counts`	Integer(s). Case counts to convert.
`pop`	Integer. Population size where cases were counted.
`digits`	Integer. Number of decimals to round to.
`rate_adj_pop`	Integer. Optional target population to use for rate. Defaults to 100k for rate per 100k.

Value

The count(s) as rates per rate_adj_pop.

Examples

```
convert_counts_to_rate(50, 200000, 2)
convert_counts_to_rate(c(10, 20), 100000, 1, 10000)
```

create_epitrax_from_file

Create an EpiTrax object from data file

Description

`create_epitrax_from_file` reads an EpiTrax data file and creates a structured object containing the data along with commonly used metadata and empty report lists.

Usage

```
create_epitrax_from_file(filepath = NULL, num_yrs = 5)
```

Arguments

`filepath`	Optional filepath. EpiTrax data file should be a CSV. If this parameter is NULL, the user will be prompted to choose a file interactively.
`num_yrs`	Integer. Number of years of data to keep. Defaults to 5.

Value

An object of class `epitrax` containing:

- `data`: The validated and formatted EpiTrax data
- `diseases`: Vector of unique diseases in the dataset
- `yrs`: Vector of years in the dataset
- `report_year`: Most recent year in the dataset
- `report_month`: Most recent month in report_year
- `internal_reports`: Empty list to store internal reports
- `public_reports`: Empty list to store public reports

See Also

`read_epitrax_data()` which this function wraps and `setup_epitrax()` which wraps this function

Examples

```
if (interactive()) {
  # Interactive file chooser:
  create_epitrax_from_file()
}

# Using sample data included with package
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
epitrax <- create_epitrax_from_file(data_file)

# Access components
head(epitrax$data)
epitrax$diseases
epitrax$report_year
```

`create_filesystem` *Create filesystem*

Description

`create_filesystem` creates the given folders if they don't already exist.

Usage

```
create_filesystem(internal, public, settings)
```

Arguments

`internal`	Filepath. Folder for internal reports.
`public`	Filepath. Folder for public reports.
`settings`	Filepath. Folder for report settings.

Value

NULL.

Examples

```
internal_folder = file.path(tempdir(), "internal")
public_folder = file.path(tempdir(), "public")
settings_folder = file.path(tempdir(), "settings")

create_filesystem(
  internal = internal_folder,
  public = public_folder,
  settings = settings_folder
)

unlink(c(internal_folder, public_folder, settings_folder), recursive = TRUE)
```

`create_public_report_combined_month_ytd` *Create combined monthly and year-to-date public report*

Description

'create_public_report_combined_month_ytd' creates a comprehensive public report that combines monthly case data with year-to-date statistics for the given month and year. This provides both current month context and cumulative year progress.

Usage

```
create_public_report_combined_month_ytd(data, diseases, y, m, config)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)
`diseases`	Dataframe. Diseases to include in the report. Maps EpiTrax disease names to public-facing versions. Must have columns: • `EpiTrax_name` (character) • `Public_name` (character)
`y`	Integer. Report year
`m`	Integer. Report month (1-12)
`config`	List. Report settings

Details

Uses the following config options:

- `current_population`
- `avg_5yr_population`
- `rounding_decimals`
- `trend_threshold`

Value

List containing the report name and combined monthly/YTD report data with columns for monthly cases/averages/trends and YTD statistics.

See Also

`create_public_report_month()`, `create_report_ytd_counts()` which this function uses and `epitraxr_config()` for config options

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
# Read in EpiTrax data
data <- read_epitrax_data(data_file)

diseases <- data.frame(
  EpiTrax_name = c("Influenza", "COVID-19", "Measles", "Syphilis"),
  Public_name = c("Influenza", "COVID-19", "Measles", "Syphilis")
)
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                          package = "epitraxr")
config <- get_report_config(config_file)
create_public_report_combined_month_ytd(
 data = data,
 diseases = diseases,
 y = 2024,
 m = 2,
 config = config
)
```

`create_public_report_month` *Create a monthly cross-section public report*

Description

'create_public_report_month' creates a public report for the given month.

Usage

```
create_public_report_month(data, diseases, y, m, config)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)
`diseases`	Dataframe. Diseases to include in the report. Maps EpiTrax disease names to public-facing versions. Must have columns: • `EpiTrax_name` (character) • `Public_name` (character)
`y`	Integer. Report year
`m`	Integer. Report month (1-12)
`config`	List. Report settings

Details

Uses the following config options:

- `current_population`
- `avg_5yr_population`
- `rounding_decimals`
- `trend_threshold`

Value

List containing the report name and data.

See Also

`get_month_counts()`, `create_report_monthly_avgs()` which this function uses and `epitraxr_config()` for config options

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
# Read in EpiTrax data
data <- read_epitrax_data(data_file)

diseases <- data.frame(
  EpiTrax_name = c("Influenza", "COVID-19", "Measles", "Syphilis"),
  Public_name = c("Influenza", "COVID-19", "Measles", "Syphilis")
)
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                          package = "epitraxr")
config <- get_report_config(config_file)

create_public_report_month(
 data = data,
 diseases = diseases,
 y = 2024,
 m = 1,
 config = config
)
```

`create_public_report_ytd`

Create a YTD public report

Description

'create_public_report_ytd' creates a public report for YTD rates.

Usage

```
create_public_report_ytd(data, diseases, y, m, config)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer)

	• `month` (integer) • `counts` (integer)
`diseases`	Dataframe. Diseases to include in the report. Maps EpiTrax disease names to public-facing versions. Must have columns: • `EpiTrax_name` (character) • `Public_name` (character)
`y`	Integer. Report year
`m`	Integer. Report month (1-12)
`config`	List. Report settings

Details

Uses the following config options:

- `current_population`
- `avg_5yr_population`
- `rounding_decimals`
- `trend_threshold`

Value

List containing the report name and data.

See Also

`create_report_ytd_counts()` which this function uses and `epitraxr_config()` for config options

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
# Read in EpiTrax data
data <- read_epitrax_data(data_file)

diseases <- data.frame(
  EpiTrax_name = c("Influenza", "COVID-19", "Measles", "Syphilis"),
  Public_name = c("Influenza", "COVID-19", "Measles", "Syphilis")
)
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                          package = "epitraxr")
config <- get_report_config(config_file)
create_public_report_ytd(
 data = data,
 diseases = diseases,
 y = 2024,
 m = 1,
 config = config
)
```

`create_report_annual_counts` *Create annual counts report*

Description

'create_report_annual_counts' generates a data frame of annual case counts for each disease, with years as columns.

Usage

```
create_report_annual_counts(data, diseases)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)
`diseases`	Character vector. Diseases to include in the report

Value

Dataframe of annual counts with one row per disease and one column per year.

Examples

```
data <- data.frame(
  disease = c("A", "A", "B"),
  year = c(2020, 2021, 2020),
  counts = c(5, 7, 8)
)
create_report_annual_counts(data, diseases = c("A", "B", "C"))
```

`create_report_grouped_stats` *Create grouped disease statistics report*

Description

'create_report_grouped_stats' generates a comprehensive report with current and historical statistics for diseases organized by group. The report includes monthly counts/rates, year-to-date counts, and trend analysis.

Usage

```
create_report_grouped_stats(data, diseases, y, m, config)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)
`diseases`	Dataframe. Diseases to include in the report. Must have column `EpiTrax_name` (character) with diseases to include. Optionally may have column `Group_name` (character) to define disease groupings. If `Group_name` is missing, all diseases will be grouped under "Uncategorized".
`y`	Integer. Report year
`m`	Integer. Report month (1-12)
`config`	List. Report settings

Details

Uses the following config options:

- `current_population`
- `avg_5yr_population`
- `rounding_decimals`
- `trend_threshold`

Value

Dataframe with one row per disease containing:

- Group: Disease group name
- Disease: Disease name
- Monthly counts and rates for current year/month
- Historical monthly averages and medians
- Year-to-date counts and historical averages and medians
- YTD trend indicators

See Also

`create_report_monthly_counts()`, `create_report_monthly_avgs()`, `create_report_monthly_medians()`, `create_report_ytd_counts()`, `create_report_ytd_medians()` which this function uses and `epitraxr_config()` for config options

Examples

```
data <- data.frame(
  disease = c("A", "A", "B", "B"),
  year = c(2023, 2024, 2023, 2024),
  month = c(1, 1, 2, 2),
  counts = c(10, 20, 15, 25)
)
diseases <- data.frame(
  EpiTrax_name = c("A", "B", "C"),
  Group_name = c("Group1", "Group1", "Group2")
)
config <- list(
  current_population = 100000,
  avg_5yr_population = 100000,
  rounding_decimals = 1,
  trend_threshold = 0.15
)
create_report_grouped_stats(data, diseases, 2024, 2, config)
```

---

create_report_monthly_avgs

Create monthly averages report

Description

'create_report_monthly_avgs' generates a data frame of average monthly case counts for each disease across all years in the input data.

Usage

```
create_report_monthly_avgs(data, diseases, config)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)
`diseases`	Character vector. Diseases to include in the report
`config`	List. Report settings

Details

Uses the following config options:

- `rounding_decimals`

Value

Dataframe of monthly averages with one row per disease and one column per month (Jan through Dec).

See Also

`epitraxr_config()` for config options

Examples

```
data <- data.frame(
  disease = c("A", "A", "B", "B"),
  year = c(2023, 2024, 2023, 2024),
  month = c(1, 1, 2, 2),
  counts = c(10, 20, 15, 25)
)
config <- list(rounding_decimals = 1)
create_report_monthly_avgs(data, c("A", "B", "C"), config)
```

`create_report_monthly_counts` *Create monthly counts report*

Description

'create_report_monthly_counts' generates a data frame of monthly case counts for each disease for a specific year, with months as columns.

Usage

```
create_report_monthly_counts(data, diseases, y)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)
`diseases`	Character vector. Diseases to include in the report
`y`	Integer. Report year

Value

Dataframe of monthly counts with one row per disease and one column per month (Jan through Dec).

Examples

```
data <- data.frame(
  disease = c("A", "A", "B", "B"),
  year = c(2024, 2024, 2024, 2023),
  month = c(1, 2, 1, 4),
  counts = c(5, 7, 8, 9)
)
create_report_monthly_counts(data, diseases = c("A", "B", "C"), y = 2024)
```

---

create_report_monthly_medians

Create monthly medians report

Description

'create_report_monthly_medians' generates a data frame of median monthly case counts for each disease across all years in the input data. This provides a more robust central tendency measure compared to averages for skewed data.

Usage

```
create_report_monthly_medians(data, diseases)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)
`diseases`	Character vector. Diseases to include in the report

Value

Dataframe of monthly medians with one row per disease and one column per month (Jan through Dec).

Examples

```
data <- data.frame(
  disease = c("A", "A", "A", "B", "B", "B"),
  year = c(2022, 2023, 2024, 2022, 2023, 2024),
  month = c(1, 1, 1, 2, 2, 2),
  counts = c(10, 20, 30, 5, 15, 25)
)
create_report_monthly_medians(data, c("A", "B", "C"))
```

create_report_ytd_counts

Create year-to-date (YTD) counts report

Description

'create_report_ytd_counts' generates a data frame of year-to-date counts for each disease up to the given month, comparing the given year to the average of other years.

Usage

```
create_report_ytd_counts(data, diseases, y, m, config, as.rates = FALSE)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)
`diseases`	Character vector. Diseases to include in the report
`y`	Integer. Report year
`m`	Integer. Report month (1-12)
`config`	List. Report settings
`as.rates`	Logical. If TRUE, returns rates per 100k instead of raw counts

Details

Uses the following config options:

- `current_population`
- `avg_5yr_population`
- `rounding_decimals`

Value

Dataframe with one row per disease and columns for current YTD and average YTD values (either counts or rates per 100k)

See Also

`epitraxr_config()` for config options

Examples

```
data <- data.frame(
  disease = c("A", "A", "B", "B"),
  year = c(2024, 2023, 2024, 2023),
  month = c(1, 1, 2, 2),
  counts = c(10, 20, 15, 25)
)
config <- list(
  current_population = 100000,
  avg_5yr_population = 100000,
  rounding_decimals = 1
)
create_report_ytd_counts(data, c("A", "B", "C"), 2024, 2, config)
```

`create_report_ytd_medians`

Create year-to-date (YTD) medians report

Description

'create_report_ytd_medians' generates a data frame of median year-to-date counts for each disease up to the given month (months `1:m`) across all years in the data. This provides a robust central tendency measure for YTD values.

Usage

```
create_report_ytd_medians(data, diseases, m)
```

Arguments

`data`	Dataframe. Input data with columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)
`diseases`	Character vector. Diseases to include in the report
`m`	Integer. Report month (1-12)

Value

Dataframe with one row per disease and columns for disease name and median YTD counts.

Examples

```
data <- data.frame(
  disease = c("A", "A", "A", "B", "B", "B"),
  year = c(2022, 2023, 2024, 2022, 2023, 2024),
  month = c(1, 1, 2, 2, 2, 3),
  counts = c(10, 15, 20, 5, 8, 12)
)
create_report_ytd_medians(data, c("A", "B", "C"), 2)
```

`epitraxr_config` *Create epitraxr config object*

Description

`epitraxr_config` creates a list of configuration options used for generating reports.

Usage

```
epitraxr_config(
  current_population = 1e+05,
  avg_5yr_population = 1e+05,
  rounding_decimals = 2,
  generate_csvs = TRUE,
  trend_threshold = 0.15
)
```

Arguments

`current_population`	Integer. Defaults to 100,000.
`avg_5yr_population`	Integer. Defaults to 100,000.
`rounding_decimals`	Integer. Defaults to 2.
`generate_csvs`	Logical. Defaults to TRUE.
`trend_threshold`	Numeric. Defaults to 0.15.

Value

A named list with 'keys' corresponding to config options.

Examples

```
epitraxr_config(
  current_population = 56000,
  avg_5yr_population = 57000,
  rounding_decimals = 3,
  generate_csvs = FALSE,
  trend_threshold = 0.2
)
```

`epitrax_ireport_annual_counts`

Create annual counts internal report from an EpiTrax object

Description

`epitrax_ireport_annual_counts` generates an internal report of annual counts for each disease in the EpiTrax object data.

Usage

```
epitrax_ireport_annual_counts(epitrax)
```

Arguments

`epitrax`	Object of class `epitrax`.

Value

Updated EpiTrax object with `annual_counts` added to the `internal_reports` field.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_ireport_annual_counts()

epitrax$internal_reports$annual_counts
```

`epitrax_ireport_monthly_avgs`
Create monthly averages internal report from an EpiTrax object

---

Description

`epitrax_ireport_monthly_avgs` generates an internal report of monthly averages for all years in the EpiTrax object data, with the option to exclude the current report year.

Usage

```
epitrax_ireport_monthly_avgs(epitrax, exclude.report.year = FALSE)
```

Arguments

`epitrax`	Object of class `epitrax`.
`exclude.report.year`	Logical indicating whether to exclude the current report year from the report. Defaults to FALSE.

Value

Updated EpiTrax object with monthly averages report added to the `internal_reports` field.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_ireport_monthly_avgs()

names(epitrax$internal_reports)
```

epitrax_ireport_monthly_counts_all_yrs

Create monthly counts internal report for all years from an EpiTrax object

Description

`epitrax_ireport_monthly_counts_all_yrs` generates internal reports of monthly counts for each year in the EpiTrax object data.

Usage

```
epitrax_ireport_monthly_counts_all_yrs(epitrax)
```

Arguments

`epitrax`	Object of class `epitrax`.

Value

Updated EpiTrax object with monthly counts reports for each year added to the `internal_reports` field.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_ireport_monthly_counts_all_yrs()

names(epitrax$internal_reports)
```

epitrax_ireport_ytd_counts_for_month

Create year-to-date (YTD) counts internal report for a given month from an EpiTrax object

Description

`epitrax_ireport_ytd_counts_for_month` generates an internal report of year-to-date counts up to a specific month in the EpiTrax object data.

Usage

```
epitrax_ireport_ytd_counts_for_month(epitrax, as.rates = FALSE)
```

Arguments

`epitrax`	Object of class `epitrax`.
`as.rates`	Logical. If TRUE, returns rates per 100k instead of raw counts.

Value

Updated EpiTrax object with report added to the `internal_reports` field.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_ireport_ytd_counts_for_month(as.rates = TRUE)

names(epitrax$internal_reports)
```

epitrax_preport_combined_month_ytd

Create combined monthly/YTD stats public report from an EpiTrax object

Description

`epitrax_preport_combined_month_ytd` generates a public report of monthly and year-to-date (YTD) disease statistics for the report month in the EpiTrax object data.

Usage

```
epitrax_preport_combined_month_ytd(epitrax)
```

Arguments

`epitrax`	Object of class `epitrax`.

Value

Updated EpiTrax object with YTD rates report added to the `public_reports` field.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_preport_combined_month_ytd()

names(epitrax$public_reports)
```

`epitrax_preport_month_crosssections` *Create monthly cross-section reports from an EpiTrax object*

Description

`epitrax_preport_month_crosssections` generates monthly cross-section reports. These compare the counts for a given month against the monthly averages for the same month across previous years.

Usage

```
epitrax_preport_month_crosssections(epitrax, month_offsets = 0:3)
```

Arguments

`epitrax`	Object of class `epitrax`.
`month_offsets`	Numeric vector of month offsets to create reports for. Defaults to 0:3, which generates reports for the current month and the three previous months.

Value

Updated EpiTrax object with monthly cross-section reports added to the `public_reports` field.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_preport_month_crosssections(month_offsets = 0:1)

names(epitrax$public_reports)
```

epitrax_preport_ytd_rates

Create year-to-date (YTD) rates public report from an EpiTrax object

Description

`epitrax_preport_ytd_rates` generates a public report of year-to-date rates for the current month in the EpiTrax object data.

Usage

```
epitrax_preport_ytd_rates(epitrax)
```

Arguments

`epitrax`	Object of class `epitrax`.

Value

Updated EpiTrax object with YTD rates report added to the `public_reports` field.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_preport_ytd_rates()

names(epitrax$public_reports)
```

epitrax_report_grouped_stats *Create grouped disease statistics report from an EpiTrax object*

Description

`epitrax_report_grouped_stats` generates a comprehensive report with current and historical statistics for diseases organized by group. The report includes monthly counts/rates, historical averages and medians, year-to-date counts, and trend analysis. It can be run for either internal or public reports.

Usage

```
epitrax_report_grouped_stats(epitrax, is.public = FALSE)
```

Arguments

`epitrax`	Object of class `epitrax`.
`is.public`	Logical indicating whether to generate a public report using the public disease list. If FALSE (default), generates an internal report using the internal disease list.

Value

Updated EpiTrax object with grouped statistics report added to either the `internal_reports` or `public_reports` field, depending on the `is.public` parameter.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_report_grouped_stats()

names(epitrax$internal_reports)
```

epitrax_report_monthly_medians
Create monthly medians report from an EpiTrax object

Description

`epitrax_report_monthly_medians` generates a report of monthly medians for all years in the EpiTrax object data, with the option to exclude the current report year. It can be run for either internal or public reports.

Usage

```
epitrax_report_monthly_medians(
  epitrax,
  is.public = FALSE,
  exclude.report.year = FALSE
)
```

Arguments

`epitrax`	Object of class `epitrax`.
`is.public`	Logical indicating whether to generate a public report using the public disease list. If FALSE (default), generates an internal report using the internal disease list.
`exclude.report.year`	Logical indicating whether to exclude the current report year from the report. Defaults to FALSE.

Value

Updated EpiTrax object with monthly medians report added to either the `internal_reports` or `public_reports` field, depending on the `is.public` parameter.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_report_monthly_medians()

names(epitrax$internal_reports)
```

epitrax_report_ytd_medians

Create year-to-date (YTD) medians report from an EpiTrax object

Description

`epitrax_report_ytd_medians` generates a report of median year-to-date counts for each disease up to the current report month across all years in the EpiTrax object data, with the option to exclude the current report year. It can be run for either internal or public reports.

Usage

```
epitrax_report_ytd_medians(
  epitrax,
  is.public = FALSE,
  exclude.report.year = FALSE
)
```

Arguments

`epitrax`	Object of class `epitrax`.
`is.public`	Logical indicating whether to generate a public report using the public disease list. If FALSE (default), generates an internal report using the internal disease list.
`exclude.report.year`	Logical indicating whether to exclude the current report year from the report. Defaults to FALSE.

Value

Updated EpiTrax object with YTD medians report added to either the `internal_reports` or `public_reports` field, depending on the `is.public` parameter.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)
```

```
epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_report_ytd_medians()

names(epitrax$internal_reports)
```

---

epitrax_set_config_from_file

Set report configuration of EpiTrax object from config file

Description

`epitrax_set_config_from_file` reads a report configuration file and adds it to the EpiTrax object.

Usage

```
epitrax_set_config_from_file(epitrax, filepath)
```

Arguments

`epitrax`	Object of class `epitrax`.
`filepath`	Path to the report configuration file.

Value

Updated EpiTrax object with `config` field set.

See Also

`epitrax_set_config_from_list()` and the convenience function `setup_epitrax()` which wraps this function

Examples

```
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
epitrax <- structure(
  list(data = c(1,2,3)),
  class = "epitrax"
)
epitrax <- epitrax_set_config_from_file(epitrax, config_file)
```

epitrax_set_config_from_list

Set report configuration of EpiTrax object from list

Description

`epitrax_set_config_from_list` sets the report configuration from the given list.

Usage

```
epitrax_set_config_from_list(epitrax, config = NULL)
```

Arguments

`epitrax`	Object of class `epitrax`.
`config`	Optional list of config parameters. If omitted, default values will be used.

Value

Updated EpiTrax object with `config` field set.

See Also

`epitrax_set_config_from_file()` and the convenience function `setup_epitrax()` which wraps this function

Examples

```
config <- list(
 current_population = 56000,
 avg_5yr_population = 57000,
 rounding_decimals = 3,
 generate_csvs = FALSE
)
epitrax <- structure(
  list(data = c(1,2,3)),
  class = "epitrax"
)
epitrax <- epitrax_set_config_from_list(epitrax, config)
```

epitrax_set_report_diseases

Set report diseases in EpiTrax object

Description

`epitrax_set_report_diseases` reads internal and public disease lists and adds them to the EpiTrax object.

Usage

```
epitrax_set_report_diseases(epitrax, disease_list_files = NULL)
```

Arguments

`epitrax`	Object of class `epitrax`.
`disease_list_files`	Optional list containing filepaths to internal and public report disease lists. If omitted, the default lists will be used and a warning will be given.

Value

Updated EpiTrax object with `report_diseases` field set.

See Also

`setup_epitrax()` the convenience function which wraps this function

Examples

```
i_file <- system.file("tinytest/test_files/disease_lists/internal_list.csv",
                      package = "epitraxr")
p_file <- system.file("tinytest/test_files/disease_lists/public_list.csv",
                      package = "epitraxr")

epitrax <- structure(
  list(data = c(1,2,3)),
  class = "epitrax"
)

epitrax <- epitrax_set_report_diseases(
  epitrax,
  disease_list_files = list(
    internal = i_file,
    public = p_file
  )
)
```

`epitrax_write_csvs` *Write reports from EpiTrax object to CSV files*

Description

`epitrax_write_csvs` writes the internal and public reports from an EpiTrax object to CSV files in the specified filesystem. Doesn't write files if the EpiTrax config setting `generate_csvs` is set to false.

Usage

```
epitrax_write_csvs(epitrax, fsys)
```

Arguments

`epitrax`	Object of class `epitrax`.
`fsys`	Filesystem list containing paths for internal and public reports.

Value

The original EpiTrax object, unchanged.

Examples

```
fsys <- list(
  internal = file.path(tempdir(), "internal_reports"),
  public = file.path(tempdir(), "public_reports"),
  settings = file.path(tempdir(), "report_settings")
)
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_preport_ytd_rates() |>
 epitrax_write_csvs(fsys = fsys)

# Cleanup
unlink(unlist(fsys, use.names = FALSE), recursive = TRUE)
```

`epitrax_write_pdf_grouped_stats` *Write grouped statistics reports from EpiTrax object to PDF files*

Description

`epitrax_write_pdf_grouped_stats` writes the grouped statistics reports from an EpiTrax object to PDF files using a formatted template. It processes both internal and public grouped statistics reports.

Usage

```
epitrax_write_pdf_grouped_stats(epitrax, params, fsys, trend.only = FALSE)
```

Arguments

`epitrax`	Object of class `epitrax`.
`params`	List. Report parameters containing: • `title`: Report title (defaults to "Grouped Report")
`fsys`	Filesystem list containing paths for internal and public reports.
`trend.only`	Logical. Whether to show only trend in the PDF report.

Value

The original EpiTrax object, unchanged.

Examples

```
## Not run:
  # Example not run because it requires LaTeX installation

  fsys <- list(
    internal = file.path(tempdir(), "internal_reports"),
    public = file.path(tempdir(), "public_reports"),
    settings = file.path(tempdir(), "report_settings")
  )
  fsys <- setup_filesystem(fsys)

  data_file <- system.file("sample_data/sample_epitrax_data.csv",
                           package = "epitraxr")
  config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                             package = "epitraxr")
  disease_lists <- list(
    internal = "use_defaults",
    public = "use_defaults"
  )

  params <- list(
    title = "Monthly Grouped Disease Statistics"
  )

  epitrax <- setup_epitrax(
    filepath = data_file,
    config_file = config_file,
    disease_list_files = disease_lists
  ) |>
    epitrax_report_grouped_stats() |>
    epitrax_write_pdf_grouped_stats(params = params, fsys = fsys)

  # Cleanup
  unlink(unlist(fsys, use.names = FALSE), recursive = TRUE)

## End(Not run)
```

`epitrax_write_pdf_public_reports` *Create formatted PDF report of monthly cross-section reports*

Description

`epitrax_write_pdf_public_reports` writes a PDF report for each public report, excluding grouped stats reports (which are handled by `epitrax_write_pdf_grouped_stats`). The PDF uses pretty formatting and adds a header and footer.

Usage

```
epitrax_write_pdf_public_reports(epitrax, fsys, trend.only = FALSE)
```

Arguments

`epitrax`	Object of class `epitrax`.
`fsys`	Filesystem list containing paths for internal and public reports.
`trend.only`	Logical. Whether to show only trend in the PDF report.

Value

The original EpiTrax object, unchanged.

Examples

```
## Not run:
  # Example not run because it requires LaTeX installation

  fsys <- list(
    internal = file.path(tempdir(), "internal_reports"),
    public = file.path(tempdir(), "public_reports"),
    settings = file.path(tempdir(), "report_settings")
  )
  fsys <- setup_filesystem(fsys)

  data_file <- system.file("sample_data/sample_epitrax_data.csv",
                           package = "epitraxr")
  config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                             package = "epitraxr")
  disease_lists <- list(
    internal = "use_defaults",
    public = "use_defaults"
  )

  epitrax <- setup_epitrax(
    filepath = data_file,
    config_file = config_file,
    disease_list_files = disease_lists
  ) |>
    epitrax_preport_month_crosssections(month_offsets = 0) |>
    epitrax_write_pdf_public_reports(fsys = fsys)

  # Cleanup
  unlink(unlist(fsys, use.names = FALSE), recursive = TRUE)

## End(Not run)
```

---

epitrax_write_xlsxs *Write reports from EpiTrax object to Excel files*

Description

`epitrax_write_xlsxs` writes the internal and public reports from an EpiTrax object to Excel files in the specified filesystem. Combines all internal reports into one Excel file with separate sheets for each report. Likewise with public reports.

Usage

```
epitrax_write_xlsxs(epitrax, fsys)
```

Arguments

`epitrax`	Object of class `epitrax`.
`fsys`	Filesystem list containing paths for internal and public reports.

Value

The original EpiTrax object, unchanged.

Examples

```
fsys <- list(
  internal = file.path(tempdir(), "internal_reports"),
  public = file.path(tempdir(), "public_reports"),
  settings = file.path(tempdir(), "report_settings")
)
fsys <- setup_filesystem(fsys)

data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
config_file <- system.file("tinytest/test_files/configs/good_config.yaml",
                           package = "epitraxr")
disease_lists <- list(
  internal = "use_defaults",
  public = "use_defaults"
)

epitrax <- setup_epitrax(
  filepath = data_file,
  config_file = config_file,
  disease_list_files = disease_lists
) |>
 epitrax_preport_ytd_rates() |>
 epitrax_write_xlsxs(fsys = fsys)

# Cleanup
unlink(unlist(fsys, use.names = FALSE), recursive = TRUE)
```

`format_epitrax_data` *Format EpiTrax data for report generation*

Description

`format_epitrax_data` prepares the input EpiTrax data for use by report generation functions in the package. It adds the `counts` column, renames columns to standard names used by the package ("disease", "month", "year", "counts"), and rearranges columns for consistency.

Usage

```
format_epitrax_data(data)
```

Arguments

`data`	Dataframe. Must contain columns: • `patient_disease` (character, unchanged from EpiTrax export) • `patient_mmwr_year` (integer, unchanged from EpiTrax export) • `month` (integer, converted from `patient_mmwr_week` by `mmwr_week_to_month()`)

Value

A standardized data frame with columns "disease", "month", "year", and "counts".

See Also

`mmwr_week_to_month()`

Examples

```
df <- data.frame(
  patient_mmwr_year = c(2020L, 2020L),
  month = c(1, 2),
  patient_disease = c("A", "B")
)
df <- format_epitrax_data(df)
```

`get_month_counts` *Get monthly counts for each disease*

Description

`get_month_counts` aggregates disease counts by month and year. This is a helper function used internally by report generation functions to summarize monthly disease counts.

Usage

```
get_month_counts(data)
```

Arguments

`data`	Dataframe. Must contain columns: • `disease` (character) • `year` (integer) • `month` (integer) • `counts` (integer)

Value

A dataframe with the aggregated monthly counts

Examples

```
df <- data.frame(
  disease = c("Flu", "Flu", "Measles"),
  year = c(2020, 2020, 2020),
  month = c(1, 1, 2),
  counts = c(5, 3, 2)
)
get_month_counts(df)
```

`get_report_config` *Read in the report config YAML file*

Description

'get_report_config' reads in the config YAML file. Missing fields will be set to default values and a warning will be issued. The config file can have the following fields:

- `current_population`: Integer. Current population size.
- `avg_5yr_population`: Integer. Average population over the last 5 years.
- `rounding_decimals`: Integer. Number of decimals to round report values to.
- `generate_csvs`: Logical. Whether to generate CSV files.
- `trend_threshold`: Numeric. Threshold for trend calculations.

Usage

```
get_report_config(filepath)
```

Arguments

`filepath`	Filepath. Path to report config file.

Details

See the example config file here: `system.file("sample_data/sample_config.yml", package = "epitraxr")`.

Value

A named list with an attribute of 'keys' from the file.

Examples

```
config_file <- system.file("sample_data/sample_config.yml",
                           package = "epitraxr")
report_config <- get_report_config(config_file)
```

get_report_diseases *Get both internal and public disease lists*

Description

get_report_diseases is a convenience function that combines get_report_diseases_internal and get_report_diseases_public.

Usage

```
get_report_diseases(internal, public, defaults)
```

Arguments

internal	Filepath. Path to internal disease list CSV file.
public	Filepath. Path to public disease list CSV file.
defaults	String vector. List of default diseases to use if either file doesn't exist.

Value

A list with two elements:

- internal: Dataframe with EpiTrax_name column
- public: Dataframe with EpiTrax_name and Public_name columns

See Also

get_report_diseases_internal(), get_report_diseases_public() which this function wraps.

Examples

```
# Using default lists (when files don't exist)
default_list <- c("Measles", "Chickenpox")
disease_lists <- get_report_diseases("", "", default_list)

# Using disease list files
i_file <- system.file("tinytest/test_files/disease_lists/internal_list.csv",
                       package = "epitraxr")
p_file <- system.file("tinytest/test_files/disease_lists/public_list.csv",
                       package = "epitraxr")
disease_lists <- get_report_diseases(
  internal = i_file,
  public = p_file,
  defaults = default_list
)
```

`get_report_diseases_internal`

Get the internal disease list

Description

'get_report_diseases_internal' reads the internal list from a given CSV file or uses the default diseases, if the file doesn't exist.

Usage

```
get_report_diseases_internal(filepath, defaults)
```

Arguments

`filepath`	Filepath. Internal disease list CSV file.
`defaults`	String vector. List of default diseases to use if the above file doesn't exist.

Details

The provided internal disease list file must contain at least a column named `EpiTrax_name` which contains EpiTrax disease names to include in the report. The file can optionally contain a column named `Group_name`, which maps the diseases in `EpiTrax_name` to a disease group. This is only used for reports that include disease groupings.

See the example file here: `system.file("sample_data/sample_disease_list.csv", package = "epitraxr")`

Value

A dataframe containing the diseases to include in the internal report and possibly the disease groupings.

Examples

```
# Using default list (when file doesn't exist)
default_list <- c("Measles", "Chickenpox")
disease_list <- get_report_diseases_internal("", default_list)

# Using a disease list file
list_file <- system.file("sample_data/sample_disease_list.csv",
                        package = "epitraxr")
disease_list <- get_report_diseases_internal(list_file, default_list)
```

get_report_diseases_public

Get the public disease list

Description

'get_report_diseases_public' reads the public list from a given CSV file or uses the default diseases if the file doesn't exist.

Usage

```
get_report_diseases_public(filepath, defaults)
```

Arguments

`filepath`	Filepath. Public disease list CSV file.
`defaults`	String vector. List of default diseases to use if the above file doesn't exist.

Details

The provided public disease list file must contain two columns named `EpiTrax_name` and `Public_name` which map EpiTrax disease names to a public-facing name for the public report. The file can optionally contain a column named `Group_name`, which maps the diseases in `EpiTrax_name` to a disease group. This is only used for reports that include disease groupings.

See the example file here: `system.file("sample_data/sample_disease_list.csv", package = "epitraxr")`

Value

A dataframe containing the diseases to include in the public report and the name to use for each disease in the public report. It may also contain the disease groupings.

Examples

```
# Using default list (when file doesn't exist)
default_list <- c("Measles", "Chickenpox")
disease_list <- get_report_diseases_public("", default_list)

# Using a disease list file
list_file <- system.file("sample_data/sample_disease_list.csv",
                         package = "epitraxr")
disease_list <- get_report_diseases_public(list_file, default_list)
```

`get_yrs` *Get unique years from the data*

Description

'get_yrs' extracts and returns the sorted unique years from the 'year' column of a data frame.

Usage

```
get_yrs(data)
```

Arguments

`data`	Dataframe. Must contain the column: • `year` (integer)

Value

Integer vector of sorted unique years present in the data.

Examples

```
df <- data.frame(year = c(2020, 2021, 2020, 2022))
get_yrs(df)
```

`mmwr_week_to_month` *Convert MMWR week to calendar month*

Description

`mmwr_week_to_month` calculates the calendar month from the `patient_mmwr_week` and `patient_mmwr_year` fields of the EpiTrax data. The result is stored in the `month` column and the `patient_mmwr_week` column is removed.

Usage

```
mmwr_week_to_month(data)
```

Arguments

`data`	Dataframe. Must contain columns: • `patient_mmwr_year` (integer) • `patient_mmwr_week` (integer)

Value

The input data frame with an added "month" column (integer 1-12) and removed `patient_mmwr_week` column.

Examples

```
df <- data.frame(
  patient_mmwr_year = 2020L,
  patient_mmwr_week = 1L,
  patient_disease = "A"
)
mmwr_week_to_month(df)
```

`read_epitrax_data` *Read in EpiTrax data*

Description

'read_epitrax_data' reads EpiTrax data from a CSV, validates, and formats it. It also filters rows older than given number of years. The input file must contain the columns:

- `patient_mmwr_year` (integer)
- `patient_mmwr_week` (integer)
- `patient_disease` (character)

Usage

```
read_epitrax_data(filepath = NULL, num_yrs = 5)
```

Arguments

`filepath`	Optional filepath. Data file should be a CSV. If this parameter is NULL, the user will be prompted to choose a file interactively.
`num_yrs`	Integer. Number of years of data to keep. Defaults to 5.

Details

See the example file here: `system.file("sample_data/sample_epitrax_data.csv", package = "epitraxr")`

Value

The validated and formatted EpiTrax data from the input file.

Examples

```
if(interactive()) {
  # Interactive file chooser:
  read_epitrax_data()
}

# Using a file path:
data <- read_epitrax_data(
  filepath = system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr"),
  num_yrs = 3
)
head(data)
```

`reshape_annual_wide` *Reshape data with each year as a separate column*

Description

'reshape_annual_wide' reshapes a given data frame with diseases for rows and years for columns.

Usage

```
reshape_annual_wide(data)
```

Arguments

`data`	Dataframe. Must have columns: • `disease` (character) • `year` (integer) • `counts` (integer)

Value

The reshaped data frame.

Examples

```
df <- data.frame(
  disease = c("A", "A", "B"),
  year = c(2020, 2021, 2020),
  counts = c(5, 7, 8)
)
reshape_annual_wide(df)
```

`reshape_monthly_wide` *Reshape data with each month as a separate column*

Description

'reshape_monthly_wide' reshapes a given data frame with diseases for rows and months for columns.

Usage

```
reshape_monthly_wide(data)
```

Arguments

`data`	Dataframe. Must contain columns: • `disease` (character) • `month` (integer) • `counts` (integer)

Value

The reshaped data frame.

Examples

```
df <- data.frame(
  disease = c("A", "B"),
  month = c(1, 2),
  counts = c(5, 6)
)
reshape_monthly_wide(df)
```

`run_app` *Launch the epitraxr Shiny Application*

Description

`run_app` launches the interactive Shiny web application for EpiTrax data analysis and report generation. The app provides a user-friendly interface for uploading data, configuring reports, and generating various types of disease surveillance reports.

Usage

```
run_app(...)
```

Arguments

`...`	Additional arguments passed to `shiny::shinyAppDir()`.

Value

Starts the execution of the app, printing the port to the console.

Examples

```
if (interactive() & requireNamespace("shiny")) {
  run_app()
}
```

`setup_epitrax` *Setup EpiTrax object with configuration and disease lists*

Description

`setup_epitrax` initializes an EpiTrax object with configuration and report disease lists. It is a convenience function that combines `create_epitrax_from_file()`, `epitrax_set_config_from_file()`, and `epitrax_set_report_diseases()`.

Usage

```
setup_epitrax(
  filepath = NULL,
  num_yrs = 5,
  disease_list_files = NULL,
  config_list = NULL,
  config_file = NULL
)
```

Arguments

`filepath`	Optional filepath. EpiTrax data file should be a CSV. If this parameter is NULL, the user will be prompted to choose a file interactively.
`num_yrs`	Integer. Number of years of data to keep. Defaults to 5.
`disease_list_files`	Optional list containing filepaths to internal and public report disease lists. If omitted, the default lists will be used and a warning will be given.
`config_list`, `config_file`	Configuration options may be specified as a list or as a path to a YAML config file, respectively. Only one can be specified at a time. If both are specified, the function will return an error. If both are omitted, the default config values will be used.

Value

An EpiTrax object with configuration and report diseases set.

See Also

`create_epitrax_from_file()`, `epitrax_set_config_from_file()`, `epitrax_set_config_from_list()`, and `epitrax_set_report_diseases()` which this function wraps.

Examples

```
data_file <- system.file("sample_data/sample_epitrax_data.csv",
                         package = "epitraxr")
disease_lists <- list(
  internal = system.file("tinytest/test_files/disease_lists/internal_list.csv",
                         package = "epitraxr"),
  public = system.file("tinytest/test_files/disease_lists/public_list.csv",
                       package = "epitraxr")
)

epitrax <- setup_epitrax(
  filepath = data_file,
  disease_list_files = disease_lists
)
```

`setup_filesystem` *Setup the report filesystem*

Description

`setup_filesystem` creates the necessary folder structure and optionally clears old reports. This is a convenience function that combines `create_filesystem` and `clear_old_reports`.

Usage

```
setup_filesystem(folders, clear.reports = FALSE)
```

Arguments

`folders`	List. Contains paths to report folders with elements: • `internal`: Folder for internal reports • `public`: Folder for public reports • `settings`: Folder for settings files
`clear.reports`	Logical. Whether to clear old reports from the internal and public folders. Defaults to FALSE.

Value

The input folders list, unchanged.

See Also

`create_filesystem()`, `clear_old_reports()` which this function wraps.

Examples

```
# Create folders in a temporary directory
folders <- list(
  internal = file.path(tempdir(), "internal"),
  public = file.path(tempdir(), "public"),
  settings = file.path(tempdir(), "settings")
)
setup_filesystem(folders)
unlink(unlist(folders, use.names = FALSE), recursive = TRUE)
```

`set_na_0` *Set NA values to 0*

Description

'set_na_0' sets NA values to 0 in a data frame.

Usage

```
set_na_0(df)
```

Arguments

`df`	Dataframe.

Value

Dataframe with NA values replaced by 0.

Examples

```
df <- data.frame(year = c(2020, NA, 2022))
set_na_0(df)
```

`standardize_report_diseases` *Standardize diseases for report*

Description

'standardize_report_diseases' removes rows from the data that shouldn't appear in the report and adds rows for diseases that should be in the report, but weren't in the input dataset. Added rows are filled with 0s.

Usage

```
standardize_report_diseases(data, diseases)
```

Arguments

`data`	Dataframe. Current report data.
`diseases`	Character vector. Diseases to include in the report.

Value

Report data with rows for all diseases to report.

Examples

```
df <- data.frame(disease=c("A","B","D"), counts=c(5,7,8))
standardize_report_diseases(df, c("A","C"))
```

`validate_config` *Validate config*

Description

`validate_config` checks the values of the given config list. If any values are missing or invalid, they are set to default values and a warning is issued.

Usage

```
validate_config(config)
```

Arguments

`config`	Named list.

Value

A named list with 'keys' corresponding to config options.

Examples

```
validate_config(config = list())
```

`validate_data` *Validate input EpiTrax data*

Description

'validate_data' checks the data for expected columns and data types, removes unneeded columns, and returns the resulting data. Missing or NA values will be removed with a warning. Valid data must include the following columns (and types):

- `patient_mmwr_year` (integer)
- `patient_mmwr_week` (integer)
- `patient_disease` (character)

Usage

```
validate_data(data)
```

Arguments

`data`	Dataframe. EpiTrax data to validate.

Value

The validated data with all unneeded columns removed.

Examples

```
df <- data.frame(
  patient_mmwr_year = 2020L,
  patient_mmwr_week = 1L,
  patient_disease = "A"
)
validate_data(df)
```

`validate_epitrax` *Validate EpiTrax object*

Description

`validate_epitrax` checks that the EpiTrax object is valid.

Usage

```
validate_epitrax(epitrax, report.check = TRUE)
```

Arguments

`epitrax`	Object of class `epitrax`.
`report.check`	Logical indicating whether to check report-related fields.

Value

NULL if valid, otherwise throws an error.

Examples

```
epitrax <- structure(
  list(
    data = c(1,2,3),
    config = list(rounding_decimals = 2, generate_csvs = TRUE),
    report_diseases = list(internal = "internal_list", public = "public_list")
  ),
  class = "epitrax"
)
validate_epitrax(epitrax, report.check = TRUE)
```

`validate_filesystem` *Validate filesystem structure*

Description

`validate_filesystem` checks that the filesystem structure is valid.

Usage

```
validate_filesystem(fsys)
```

Arguments

`fsys` List. Contains paths to report folders with elements:

- `internal`: Folder for internal reports
- `public`: Folder for public reports

Value

NULL if valid, otherwise throws an error.

Examples

```
fsys <- list(
  internal = "internal_reports",
  public = "public_reports"
)
validate_filesystem(fsys)
```

`write_report_csv` *Write report CSV files*

Description

`write_report_csv` writes the given data to the specified folder with the given filename.

Usage

```
write_report_csv(data, filename, folder)
```

Arguments

`data`	Dataframe. Report data.
`filename`	String. Report filename.
`folder`	Filepath. Report destination folder.

Value

NULL.

Examples

```
# Create sample data
r_data <- data.frame(
  Disease = c("Measles", "Chickenpox"),
  Counts = c(20, 43)
)

# Write to temporary directory
write_report_csv(r_data, "report.csv", tempdir())
unlink(file.path(tempdir(), "report.csv"), recursive = TRUE)
```

`write_report_pdf` *Write general PDF report of disease stats from R Markdown template*

Description

`write_report_pdf` renders a report as a PDF using a R Markdown template. It is relatively flexible and can be used for various types of report.

Usage

```
write_report_pdf(data, params, filename, folder, trend.only = FALSE)
```

Arguments

`data`	Dataframe. Report data.
`params`	List. Report parameters containing: • `title`: Report title (defaults to "Disease Report") • `report_year`: Report year (defaults to 2025) • `report_month`: Report month (defaults to 1) • `trend_threshold`: Threshold for trend calculations (defaults to 0.15)
`filename`	String. Report filename.
`folder`	Filepath. Report destination folder.
`trend.only`	Logical. Whether to show only trend in the PDF report. If TRUE, "trend_only_" will be prepended to the filename.

Value

NULL (called for side effects - creates the report file).

Examples

```
## Not run:
  # Example not run because it requires LaTeX installation

  # Create sample report data
  r_data <- data.frame(
    Disease = c("COVID", "Flu", "Measles"),
    `March 2024` = c(0, 25, 5),
    `Historical March Avg` = c(0, 15, 8),
    `Trend` = compute_trend(c(0, 25, 5), c(0, 15, 8)),
    check.names = FALSE
  )

  # Set report parameters
  params <- list(
    title = "Monthly Disease Surveillance Report",
    report_year = 2024,
    report_month = 3,
    trend_threshold = 0.20
  )

  # Write to temporary directory
  write_report_pdf(
    data = r_data,
    params = params,
    filename = "monthly_disease_report.pdf",
    folder = tempdir()
  )

## End(Not run)
```

write_report_pdf_grouped

Write PDF grouped report from R Markdown template

Description

write_report_pdf_grouped renders a grouped disease statistics report as a PDF using a R Markdown template. The report includes comprehensive disease statistics organized by groups with current and historical data.

Usage

```
write_report_pdf_grouped(data, params, filename, folder, trend.only = FALSE)
```

Arguments

data	Dataframe. Report data.
params	List. Report parameters containing: • title: Report title (defaults to "Grouped Report") • report_year: Report year (defaults to 2025) • report_month: Report month (defaults to 1) • trend_threshold: Threshold for trend calculations (defaults to 0.15)
filename	String. Report filename.
folder	Filepath. Report destination folder.
trend.only	Logical. Whether to show only trend in the PDF report. If TRUE, "trend_only_" will be prepended to the filename.

Value

NULL (called for side effects - creates the report file).

Examples

```
## Not run:
  # Example not run because it requires LaTeX installation

  # Create sample grouped report data
  r_data <- data.frame(
    Group = c("Respiratory", "Respiratory", "Vaccine-Preventable"),
    Disease = c("COVID", "Flu", "Measles"),
    `March 2024` = c(0, 25, 5),
    `March 2024 Rate` = c(0, 25, 5),
    `Historical March Avg` = c(0, 15, 8),
    `Historical March Median` = c(0, 15, 8),
    `2024 YTD` = c(0, 37, 9),
    `Historical 2024 YTD Avg` = c(20, 25, 14),
    `Historical 2024 YTD Median` = c(20, 25, 14),
    `YTD Trend` = compute_trend(c(0, 37, 9), c(20, 25, 14)),
    check.names = FALSE
  )

  # Set report parameters
  params <- list(
    title = "Grouped Disease Surveillance Report",
    report_year = 2024,
    report_month = 3,
    trend_threshold = 0.20
  )

  # Write to temporary directory
  write_report_pdf_grouped(
    data = r_data,
    params = params,
    filename = "grouped_disease_report.pdf",
    folder = tempdir()
  )

## End(Not run)
```

`write_report_xlsx` *Write report Excel files*

Description

`write_report_xlsx` writes the given data to the specified folder with the given filename in Excel (.xlsx) format.

Usage

```
write_report_xlsx(data, filename, folder)
```

Arguments

`data`	List. Named list of dataframes. The name will be used as the sheet name.
`filename`	String. Report filename.
`folder`	Filepath. Report destination folder.

Value

NULL.

Examples

```
# Create sample data with multiple sheets
r_data1 <- data.frame(
  Disease = c("Measles", "Chickenpox"),
  Counts = c(20, 43)
)
r_data2 <- data.frame(
  Disease = c("Measles", "Chickenpox"),
  Rate = c(10.5, 22.7)
)
r_xl <- list(
  counts = r_data1,
  rates = r_data2
)

# Write to temporary directory
write_report_xlsx(r_xl, "report.xlsx", tempdir())
unlink(file.path(tempdir(), "report.xlsx"), recursive = TRUE)
```

Index