

# Package ‘MetaRVM’

January 9, 2026

**Title** Meta-Population Compartmental Model for Respiratory Virus Diseases

**Version** 1.0.1

**Description** Simulates respiratory virus epidemics using meta-population compartmental models following Fadikar et. al. (2025) <[doi:10.1101/2025.05.05.25327021](https://doi.org/10.1101/2025.05.05.25327021)>. 'MetaRVM' implements a stochastic SEIRD (Susceptible-Exposed-Infected-Recovered-Dead) framework with demographic stratification by age, race, and geographic zones. It supports complex epidemiological scenarios including asymptomatic and presymptomatic transmission, hospitalization dynamics, vaccination schedules, and time-varying contact patterns via mixing matrices.

**URL** <https://RESUME-Epi.github.io/MetaRVM/>,  
<https://github.com/RESUME-Epi/MetaRVM>

**BugReports** <https://github.com/RESUME-Epi/MetaRVM/issues>

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Depends** R (>= 3.5.0)

**Imports** data.table, dplyr, ggplot2, magrittr, methods, odin, purrr,  
R6, tidyr, yaml

**Suggests** dde, knitr, pkgdown, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr, rmarkdown

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Arindam Fadikar [aut, cre, cph] (ORCID:  
<<https://orcid.org/0000-0001-7396-0350>>),  
Charles Macal [ctb],  
Martinez Moyano Ignacio Javier [ctb],  
Ozik Jonathan [ctb]

**Maintainer** Arindam Fadikar <afadikar@anl.gov>

**Repository** CRAN

**Date/Publication** 2026-01-09 17:40:02 UTC

## Contents

format_metarvm_output . . . . .	2
metaRVM . . . . .	3
MetaRVMCheck . . . . .	5
MetaRVMConfig . . . . .	6
MetaRVMResults . . . . .	9
MetaRVMSummary . . . . .	11
meta_sim . . . . .	14
parse_config . . . . .	20
<b>Index</b>	<b>23</b>

---

format\_metarvm\_output *Format MetaRVM simulation output*

---

### Description

This function formats raw MetaRVM simulation output by:

1. Converting time steps to calendar dates
2. Adding demographic attributes from population mapping
3. Handling different disease states appropriately:
  - Regular states (S, E, I, etc.): Keep values at integer time points
  - New count states (n\_ prefix): Sum pairs to get daily counts

### Usage

```
format_metarvm_output(sim_output, config)
```

### Arguments

sim_output	data.table containing raw simulation output from <code>meta_sim</code>
config	MetaRVMConfig object or config list containing parameters

### Value

data.table with formatted output including calendar dates and demographics

### Note

This function is used for formatting the `meta_sim` output when `MetaRVM` function is called.

---

`metaRVM`*Run a MetaRVM epidemic simulation*

---

## Description

`metaRVM()` is the high-level entry point for running a MetaRVM metapopulation respiratory virus simulation. It parses the configuration, runs one or more simulation instances (deterministic or stochastic), formats the ODIN/MetaRVM output into a tidy long table with calendar dates and demographic attributes, and returns a [MetaRVMResults](#) object for downstream analysis and plotting.

## Usage

```
metaRVM(config_input)
```

## Arguments

`config_input` Configuration specification in one of three forms:

- **Character string:** path to a YAML configuration file.
- **MetaRVMConfig object:** pre-initialized configuration.
- **Named list:** output from `parse_config()` with `return_object = FALSE`.

## Details

The configuration input controls:

- **Population structure** (e.g., age, race, zone)
- **Disease parameters** (`ts`, `tv`, `ve`, `de`, `dp`, `da`, `ds`, `dh`, `dr`, `pea`, `psr`, `phr`, `dv`, etc.)
- **Mixing matrices** (weekday/weekend, day/night contact patterns)
- **Vaccination schedule** and immunity waning
- **Simulation settings** (start date, length, number of instances, stochastic vs. deterministic mode, checkpointing)

Internally, `metaRVM()`:

1. Parses the YAML configuration via `parse_config()`.
2. Calls the ODIN-based simulation engine `meta_sim()` for each instance.
3. Uses `format_metarvm_output()` to convert time steps to dates and attach demographic attributes.
4. Wraps the formatted output and metadata in a [MetaRVMResults](#) object that supports method chaining for subsetting, summarizing, and plotting.

**Value**

A [MetaRVMResults](#) R6 object with three key components:

**\$results** A tidy `data.table` with one row per date–subpopulation–disease state–instance combination. Typical columns include:

- `date`: calendar date (`Date`)
- `age`, `race`, `zone`: demographic categories (if present in the population mapping)
- `disease_state`: compartment or flow label (e.g., `S`, `E`, `I_symp`, `H`, `R`, `D`, `n_SE`, `n_IsympH`, etc.)
- `value`: population count or daily flow
- `instance`: simulation instance index (1, 2, ...)

**\$config** The [MetaRVMConfig](#) object used for the run.

**\$run\_info** A list with metadata such as `n_instances`, `date_range`, `delta_t`, and checkpoint information.

**Author(s)**

Arindam Fadikar, Charles Macal, Ignacio Martinez-Moyano, Jonathan Ozik

**References**

Fadikar, A., et al. "Developing and deploying a use-inspired metapopulation modeling framework for detailed tracking of stratified health outcomes"

**See Also**

[parse\\_config\(\)](#) for reading YAML configurations, [MetaRVMConfig](#) for configuration management, [MetaRVMResults](#) for analysis and plotting, [meta\\_sim\(\)](#) for the low-level simulation engine.

**Examples**

```
options(odin.verbose = FALSE)
example_config <- system.file("extdata", "example_config.yaml",
                             package = "MetaRVM")

# Run a single-instance simulation from a YAML file
results <- metaRVM(example_config)

# Print a high-level summary
results

# Access the tidy results table
head(results$results)

# Summarize and plot hospitalizations and deaths by age and race
results$summarize(
  group_by      = c("age", "race"),
  disease_states = c("H", "D"),
  stats         = c("median", "quantile"),
```

```
    quantiles      = c(0.25, 0.75)
  )$plot()

# Using a pre-parsed configuration object
cfg <- parse_config(example_config, return_object = TRUE)
results2 <- metaRVM(cfg)
```

---

MetaRVMCheck                      *MetaRVM Checkpoint Class*

---

## Description

R6 class to handle MetaRVM checkpoint data. This class is a simplified version of [MetaRVMConfig](#) tailored for storing and accessing simulation checkpoints.

## Details

The MetaRVMCheck class is designed to hold the state of a simulation at a specific time point, allowing for continuation or analysis. It stores all necessary parameters and population states.

## Super class

[MetaRVM::MetaRVMConfig](#) -> MetaRVMCheck

## Public fields

check\_data List containing all parsed checkpoint data

## Methods

### Public methods:

- [MetaRVMCheck\\$new\(\)](#)
- [MetaRVMCheck\\$clone\(\)](#)

**Method** `new()`: Initialize a new MetaRVMCheck object

*Usage:*

```
MetaRVMCheck$new(input)
```

*Arguments:*

input A list containing checkpoint data.

*Returns:* A new MetaRVMCheck object.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
MetaRVMCheck$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Author(s)**

Arindam Fadikar

---

`MetaRVMConfig`*MetaRVM Configuration Class*

---

**Description**

R6 class to handle MetaRVM configuration data with validation and methods. This class encapsulates all configuration parameters needed for MetaRVM simulations, providing methods for parameter access, validation, and introspection.

**Details**

The MetaRVMConfig class stores parsed configuration data from YAML files and provides structured access to simulation parameters. It automatically validates configuration completeness and provides convenient methods for accessing demographic categories, population mappings, and other simulation settings.

**Public fields**

`config_file` Path to the original YAML config file (if applicable)

`config_data` List containing all parsed configuration parameters

**Methods****Public methods:**

- `MetaRVMConfig$new()`
- `MetaRVMConfig$get()`
- `MetaRVMConfig$get_all()`
- `MetaRVMConfig$list_parameters()`
- `MetaRVMConfig$parameter_summary()`
- `MetaRVMConfig$set()`
- `MetaRVMConfig$print()`
- `MetaRVMConfig$get_pop_map()`
- `MetaRVMConfig$get_age_categories()`
- `MetaRVMConfig$get_race_categories()`
- `MetaRVMConfig$get_zones()`
- `MetaRVMConfig$clone()`

**Method** `new()`: Initialize a new MetaRVMConfig object

*Usage:*

```
MetaRVMConfig$new(input)
```

*Arguments:*

input Either a file path (character) or parsed config list

*Returns:* New MetaRVMConfig object (invisible)

**Method** `get()`: Get a configuration parameter

*Usage:*

```
MetaRVMConfig$get(param)
```

*Arguments:*

param Parameter name

*Returns:* The requested parameter value

**Method** `get_all()`: Get all configuration parameters as a list

*Usage:*

```
MetaRVMConfig$get_all()
```

*Returns:* Named list of all configuration parameters

**Method** `list_parameters()`: List all available parameter names

*Usage:*

```
MetaRVMConfig$list_parameters()
```

*Returns:* Character vector of parameter names

**Method** `parameter_summary()`: Show summary of parameter types and sizes

*Usage:*

```
MetaRVMConfig$parameter_summary()
```

*Returns:* Data frame with parameter information

**Method** `set()`: Set a configuration parameter

*Usage:*

```
MetaRVMConfig$set(param, value)
```

*Arguments:*

param Character string. Parameter name to set

value The value to assign to the parameter

*Returns:* Self (invisible) for method chaining

**Method** `print()`: Print summary of configuration

*Usage:*

```
MetaRVMConfig$print()
```

*Returns:* Self (invisible)

**Method** `get_pop_map()`: Get population mapping data

*Usage:*

```
MetaRVMConfig$get_pop_map()
```

*Returns:* data.table containing population mapping with demographic categories

**Method** `get_age_categories()`: Get available age categories

*Usage:*

```
MetaRVMConfig$get_age_categories()
```

*Returns:* Character vector of unique age categories, or NULL if no population mapping available

**Method** `get_race_categories()`: Get available race categories

*Usage:*

```
MetaRVMConfig$get_race_categories()
```

*Returns:* Character vector of unique race categories, or NULL if no population mapping available

**Method** `get_zones()`: Get available zones

*Usage:*

```
MetaRVMConfig$get_zones()
```

*Returns:* Character vector of unique zone identifiers, or NULL if no population mapping available

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
MetaRVMConfig$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Author(s)

Arindam Fadikar

## Examples

```
# Initialize from YAML file
example_config <- system.file("extdata", "example_config.yaml", package = "MetaRVM")
config <- MetaRVMConfig$new(example_config)

# Access parameters
config$get("N_pop")
config$get("start_date")

# Get demographic categories
ages <- config$get_age_categories()
races <- config$get_race_categories()
zones <- config$get_zones()
```

---

MetaRVMResults	<i>MetaRVM Results Class</i>
----------------	------------------------------

---

## Description

R6 class to handle MetaRVM simulation results with comprehensive analysis and visualization methods. This class stores formatted simulation results and provides methods for data summarization, subsetting, and visualization with flexible demographic groupings.

## Details

The MetaRVMResults class automatically formats raw simulation output upon initialization, converting time steps to calendar dates and adding demographic attributes. It provides methods for flexible data summarization across any combination of age, race, and geographic zone categories, plus method chaining for streamlined analysis workflows.

## Public fields

`config` MetaRVMConfig object used to generate these results  
`results` data.table containing formatted simulation results  
`run_info` List containing run metadata

## Methods

### Public methods:

- [MetaRVMResults\\$new\(\)](#)
- [MetaRVMResults\\$print\(\)](#)
- [MetaRVMResults\\$subset\\_data\(\)](#)
- [MetaRVMResults\\$summarize\(\)](#)
- [MetaRVMResults\\$clone\(\)](#)

**Method** `new()`: Initialize a new MetaRVMResults object

*Usage:*

```
MetaRVMResults$new(  
  raw_results,  
  config,  
  run_info = NULL,  
  formatted_results = NULL  
)
```

*Arguments:*

`raw_results` Raw simulation results data.table  
`config` MetaRVMConfig object used for the simulation  
`run_info` Optional metadata about the run  
`formatted_results` formatted simulation results data.table

*Returns:* New MetaRVMResults object (invisible)

**Method** print(): Print summary of results

*Usage:*

```
MetaRVMResults$print()
```

*Returns:* Self (invisible)

**Method** subset\_data(): Subset the data based on any combination of parameters

*Usage:*

```
MetaRVMResults$subset_data(
  ages = NULL,
  races = NULL,
  zones = NULL,
  disease_states = NULL,
  date_range = NULL,
  instances = NULL,
  exclude_p_columns = TRUE
)
```

*Arguments:*

ages Vector of age categories to include (default: all)

races Vector of race categories to include (default: all)

zones Vector of zones to include (default: all)

disease\_states Vector of disease states to include (default: all, excludes p\_ columns)

date\_range Vector of two dates start\_date, and end\_date for filtering (default: all)

instances Vector of instance numbers to include (default: all)

exclude\_p\_columns Logical, whether to exclude p\_ columns (default: TRUE)

*Returns:* MetaRVMResults object with subset of results

**Method** summarize(): Summarize results across specified demographic characteristics

*Usage:*

```
MetaRVMResults$summarize(
  group_by,
  disease_states = NULL,
  date_range = NULL,
  stats = c("mean", "median", "sd"),
  quantiles = c(0.25, 0.75),
  exclude_p_columns = TRUE
)
```

*Arguments:*

group\_by Vector of demographic variables to group by: c("age", "race", "zone")

disease\_states Vector of disease states to include (default: all, excludes p\_ columns)

date\_range Optional date range for filtering

stats Vector of statistics to calculate: c("mean", "median", "sd", "min", "max", "sum", "quantile"). If NULL, returns all instances

quantiles Vector of quantiles to calculate if "quantile" is in stats (default: c(0.25, 0.75))  
 exclude\_p\_columns Logical, whether to exclude p\_ columns (default: TRUE)

*Returns:* data.table with summarized time series data or all instances if stats = NULL

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
MetaRVMResults$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

### Author(s)

Arindam Fadikar

### Examples

```
options(odin.verbose = FALSE)
example_config <- system.file("extdata", "example_config.yaml", package = "MetaRVM")
# Run simulation
results_obj <- metaRVM(example_config)
# Access formatted results
head(results_obj$results)

# Subset data with multiple filters
subset_data <- results_obj$subset_data(
  age = c("18-49", "50-64"),
  disease_state = c("H", "D"),
  date_range = c(as.Date("2024-01-01"), as.Date("2024-02-01"))
)

# Method chaining for analysis and visualization
results_obj$subset_data(disease_state = "H")$summarize(
  group_by = c("age", "race"),
  stats = c("median", "quantile"),
  quantiles = c(0.25, 0.75)
)$plot()
```

### Description

R6 class for summarized MetaRVM results with plotting capabilities and method chaining support. This class stores summarized simulation data and provides visualization methods that automatically adapt based on the data structure and grouping variables.

## Details

The MetaRVMSummary class is designed to work seamlessly with method chaining from MetaRVM-Results. It stores either summary statistics (mean, median, quantiles, etc.) or individual instance data, and provides intelligent plotting methods that automatically determine appropriate visualizations based on the data structure and demographic groupings.

The class supports two data types:

- **Summary data:** Contains aggregated statistics across simulation instances
- **Instance data:** Contains individual trajectory data for each simulation instance

Plotting behavior adapts automatically:

- Single grouping variable: Facets by demographic category, colors by disease state
- Two grouping variables: Grid layout with both demographics as facet dimensions
- Three grouping variables: Grid layout with first two as facets, third as color

## Public Fields

`data` `data.table` containing summarized results  
`config` MetaRVMConfig object from original simulation  
`type` Character string indicating data type ("summary" or "instances")

## Public fields

`data` Summarized data  
`config` Original MetaRVMConfig object  
`type` Type of summary ("instances" or "summary")

## Methods

### Public methods:

- [MetaRVMSummary\\$new\(\)](#)
- [MetaRVMSummary\\$print\(\)](#)
- [MetaRVMSummary\\$plot\(\)](#)
- [MetaRVMSummary\\$clone\(\)](#)

**Method** `new()`: Initialize MetaRVMSummary object

*Usage:*

```
MetaRVMSummary$new(data, config, type)
```

*Arguments:*

`data` `data.table` containing summarized or instance data  
`config` MetaRVMConfig object from original simulation  
`type` Character string indicating data type ("summary" or "instances")

*Returns:* New MetaRVMSummary object (invisible)

**Method** `print()`: Print summary of the data object

*Usage:*

```
MetaRVMSummary$print()
```

*Returns:* Self (invisible)

**Method** `plot()`: Plot method that shows median with quantile bands

*Usage:*

```
MetaRVMSummary$plot(ci_level = 0.95, theme = theme_minimal(), title = NULL)
```

*Arguments:*

`ci_level` Confidence level for empirical quantiles (default: 0.95). Only used if quantile columns are not pre-specified

`theme` ggplot2 theme function (default: theme\_minimal())

`title` Optional custom plot title

*Details:* This method creates time series plots with automatic layout adaptation based on grouping variables:

- For summary data: Shows median lines with quantile confidence bands
- Automatically determines faceting strategy based on number of grouping variables
- Uses disease states for color differentiation when appropriate

The method requires specific data structure:

- Summary data must contain 'median\_value' and quantile columns (e.g., 'q25', 'q75')
- Instance data must contain 'instance' column for individual trajectory grouping

*Returns:* ggplot object

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
MetaRVMSummary$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Author(s)

Arindam Fadikar

## Examples

```
options(odin.verbose = FALSE)
example_config <- system.file("extdata", "example_config_dist.yaml", package = "MetaRVM")
# Run simulation
results <- metaRVM(example_config)
# Typically created through method chaining
summary_obj <- results$subset_data(disease_state = "H")$summarize(
  group_by = c("age", "race"),
  stats = c("median", "quantile"),
  quantiles = c(0.25, 0.75)
)
```

```
# Direct plotting
summary_obj$plot()

# Plot with custom ggplot theme and confidence level
summary_obj$plot(theme = ggplot2::theme_bw())
```

---

 meta\_sim

---

*Metapopulation Respiratory Virus Model Simulator*


---

## Description

The core simulation engine that implements a stochastic compartmental SEIRD (Susceptible-Exposed-Infected-Recovered-Dead) model for respiratory virus epidemics across multiple demographic sub-populations. The function compiles and executes an ODIN-based differential equation model with time-varying contact patterns, vaccination dynamics, and complex disease progression pathways.

## Usage

```
meta_sim(
  N_pop,
  ts,
  tv,
  S0,
  I0,
  P0,
  R0,
  H0 = rep(0, N_pop),
  D0 = rep(0, N_pop),
  Ia0 = rep(0, N_pop),
  Ip0 = rep(0, N_pop),
  E0 = rep(0, N_pop),
  V0 = rep(0, N_pop),
  m_weekday_day,
  m_weekday_night,
  m_weekend_day,
  m_weekend_night,
  start_day = 0,
  delta_t,
  vac_mat,
  dv,
  de,
  pea,
  dp,
  da,
  ds,
```

```

    psr,
    dh,
    phr,
    dr,
    ve,
    nsteps,
    is.stoch = FALSE,
    seed = NULL,
    do_chk = FALSE,
    chk_time_steps = NULL,
    chk_file_names = NULL
)

```

### Arguments

N_pop	Integer. Number of demographic subpopulations in the model
ts	Numeric vector or scalar. Transmission rate for symptomatic individuals in susceptible population. If scalar, applied to all subpopulations
tv	Numeric vector or scalar. Transmission rate for symptomatic individuals in vaccinated population. If scalar, applied to all subpopulations
S0	Numeric vector of length N_pop. Initial number of susceptible individuals in each subpopulation
I0	Numeric vector of length N_pop. Initial number of symptomatic infected individuals in each subpopulation
P0	Numeric vector of length N_pop. Total population sizes for each subpopulation
R0	Numeric vector of length N_pop. Initial number of recovered individuals in each subpopulation
H0	Numeric vector of length N_pop. Initial number of hospitalized individuals in each subpopulation (default: rep(0, N_pop))
D0	Numeric vector of length N_pop. Initial number of deceased individuals in each subpopulation (default: rep(0, N_pop))
Ia0	Numeric vector of length N_pop. Initial number of asymptomatic infected individuals in each subpopulation (default: rep(0, N_pop))
Ip0	Numeric vector of length N_pop. Initial number of presymptomatic infected individuals in each subpopulation (default: rep(0, N_pop))
E0	Numeric vector of length N_pop. Initial number of exposed individuals in each subpopulation (default: rep(0, N_pop))
V0	Numeric vector of length N_pop. Initial number of vaccinated individuals in each subpopulation
m_weekday_day	Numeric matrix (N_pop × N_pop). Contact mixing matrix for weekday daytime (6 AM - 6 PM) interactions
m_weekday_night	Numeric matrix (N_pop × N_pop). Contact mixing matrix for weekday nighttime (6 PM - 6 AM) interactions

m_weekend_day	Numeric matrix ( $N_{\text{pop}} \times N_{\text{pop}}$ ). Contact mixing matrix for weekend daytime (6 AM - 6 PM) interactions
m_weekend_night	Numeric matrix ( $N_{\text{pop}} \times N_{\text{pop}}$ ). Contact mixing matrix for weekend nighttime (6 PM - 6 AM) interactions
start_day	Start day of the week expressed as an integer value between 0 and 6, 0 being Monday. Default simulation start day is Monday.
delta_t	Positive numeric. Discrete time increment in days (typically 0.5)
vac_mat	Numeric matrix. Vaccination schedule with dimensions ( $n\text{steps} \times (1 + N_{\text{pop}})$ ). First column contains time indices, remaining columns contain vaccination counts for each subpopulation at each time step
dv	Numeric vector or scalar. Mean duration (days) in vaccinated state before immunity waning. If scalar, applied to all subpopulations
de	Numeric vector or scalar. Mean duration (days) in exposed state. If scalar, applied to all subpopulations
pea	Numeric vector or scalar. Proportion of exposed individuals becoming asymptomatic infectious (vs. presymptomatic), values between 0 and 1. If scalar, applied to all subpopulations. If scalar, applied to all subpopulations
dp	Numeric vector or scalar. Mean duration (days) in presymptomatic infectious state. If scalar, applied to all subpopulations
da	Numeric vector or scalar. Mean duration (days) in asymptomatic infectious state. If scalar, applied to all subpopulations
ds	Numeric vector or scalar. Mean duration (days) in symptomatic infectious state. If scalar, applied to all subpopulations
psr	Numeric vector or scalar. Proportion of symptomatic individuals recovering directly (vs. hospitalization), values between 0 and 1. If scalar, applied to all subpopulations. If scalar, applied to all subpopulations
dh	Numeric vector or scalar. Mean duration (days) in hospitalized state. If scalar, applied to all subpopulations
phr	Numeric vector or scalar. Proportion of hospitalized individuals recovering (vs. death), values between 0 and 1. If scalar, applied to all subpopulations.
dr	Numeric vector or scalar. Mean duration (days) of immunity in recovered state. If scalar, applied to all subpopulations
ve	Numeric vector or scalar. Vaccine effectiveness (proportion), values between 0 and 1. If scalar, applied to all subpopulations
nsteps	Integer. Total number of discrete time evolution steps in simulation
is.stoch	Logical. Whether to run stochastic simulation (TRUE) or deterministic simulation (FALSE). Default: FALSE
seed	Integer or NULL. Random seed for reproducibility. Only used when is.stoch = TRUE. Default: NULL
do_chk	Logical. Whether to save model checkpoint at simulation end. Default: FALSE
chk_time_steps	Integer vector or NULL. Time steps at which to save checkpoints.
chk_file_names	List of character vectors or NULL. File names for checkpoints. Each element of the list corresponds to a time step in chk_time_steps.

## Details

The model implements a metapopulation epidemiological framework with the following features:

### Compartmental Structure:

- **S**: Susceptible individuals
- **E**: Exposed (incubating) individuals
- **I\_presymp**: Presymptomatic infectious individuals
- **I\_asymp**: Asymptomatic infectious individuals
- **I\_symp**: Symptomatic infectious individuals
- **H**: Hospitalized individuals
- **R**: Recovered individuals
- **D**: Deceased individuals
- **V**: Vaccinated individuals
- **P**: Total living population (excludes deaths)

### Disease Progression Pathways:

1. **S** → **E**: Exposure through contact with infectious individuals
2. **E** → **I\_asymp/I\_presymp**: Progression to infectious states (proportion  $p_{ea}$ )
3. **I\_presymp** → **I\_symp**: Development of symptoms
4. **I\_asymp** → **R**: Direct recovery from asymptomatic state
5. **I\_symp** → **R/H**: Recovery or hospitalization (proportion  $p_{sr}$ )
6. **H** → **R/D**: Hospital discharge or death (proportion  $p_{hr}$ )
7. **R** → **S**: Loss of immunity
8. **S** → **V**: Vaccination
9. **V** → **S/E**: Vaccine waning or breakthrough infection

**Mixing Patterns:** Contact patterns vary by:

- Day of week: Weekday vs. weekend patterns
- Time of day: Day (6 AM - 6 PM) vs. night (6 PM - 6 AM) patterns
- Each pattern specified by  $N_{pop} \times N_{pop}$  contact matrix

**Force of Infection:** Transmission occurs through contact between susceptible/vaccinated individuals and all infectious compartments ( $I_{presymp} + I_{asymp} + I_{symp}$ ), modified by:

- Population-specific transmission rates ( $t_s, t_v$ )
- Time-varying contact patterns
- Vaccine effectiveness for breakthrough infections

### Stochastic vs. Deterministic Mode:

- **Deterministic:** Uses exact differential equations
- **Stochastic:** Adds demographic stochasticity via binomial draws

**Vaccination Implementation:** Vaccination is implemented as time-varying input with:

- Scheduled vaccination counts per time step and subpopulation
- Vaccine effectiveness reducing infection probability
- Waning immunity returning individuals to susceptible state

### Value

Returns a `data.table` with the following structure:

**step** Integer time step index (0 to `nsteps`)

**time** Continuous simulation time (`step × delta_t`)

**disease\_state** Character vector of compartment names

**population\_id** Character vector of subpopulation identifiers

**value** Numeric values representing population counts in each compartment

Available disease states in output:

- Core compartments: S, E, I\_presymp, I\_asymp, I\_symp, H, R, D, V, P
- Derived outputs: I\_all (total infectious), cum\_V (cumulative vaccinations)
- Transition flows: n\_SE, n\_EI, n\_HR, n\_HD, etc. (new infections, hospitalizations, deaths)
- Debug outputs: p\_SE, p\_VE, I\_eff (probabilities and effective populations)

### Parameter Scaling

All duration parameters are automatically converted to rates ( $1/\text{duration}$ ). Scalar parameters are automatically expanded to vectors of length `N_pop`. This allows flexible specification of homogeneous or heterogeneous parameters.

### Checkpointing

When `do_chk = TRUE`, the function saves a checkpoint file containing:

- Final compartment states for simulation continuation
- All model parameters for reproducibility
- Vaccination schedule data
- Population structure information

### Author(s)

Arindam Fadikar, Charles Macal, Ignacio Martinez-Moyano, Jonathan Ozik

### References

- ODIN package: <https://mrc-ide.github.io/odin/>
- Fadikar, A., et al. "Developing and deploying a use-inspired metapopulation modeling framework for detailed tracking of stratified health outcomes"

**See Also**

[metaRVM](#) for high-level simulation interface with configuration files [parse\\_config](#) for configuration file processing [format\\_metarvm\\_output](#) for output formatting with demographics

**Examples**

```
options(odin.verbose = FALSE)
# Basic deterministic simulation
N_pop <- 2
nsteps <- 400

# Initialize populations
S0 <- rep(1000, N_pop)
I0 <- rep(10, N_pop)
P0 <- S0 + I0
R0 <- rep(0, N_pop)

# Contact matrices (simplified - identity matrices)
contact_matrix <- diag(N_pop)

# Basic vaccination schedule (10% vaccination)
vac_mat <- matrix(0, nrow = nsteps + 1, ncol = N_pop + 1)
vac_mat[, 1] <- 0:nsteps
vac_mat[1, 1 + (1:N_pop)] <- P0 * 0.1

# Run simulation
results <- meta_sim(
  N_pop = N_pop,
  ts = 0.5,
  tv = 0.1,
  S0 = S0,
  I0 = I0,
  P0 = P0,
  R0 = R0,
  m_weekday_day = contact_matrix,
  m_weekday_night = contact_matrix,
  m_weekend_day = contact_matrix,
  m_weekend_night = contact_matrix,
  delta_t = 0.5,
  vac_mat = vac_mat,
  dv = 365,
  de = 3,
  pea = 0.3,
  dp = 2,
  da = 7,
  ds = 7,
  psr = 0.95,
  dh = 10,
  phr = 0.9,
  dr = 180,
  ve = 0.8,
  nsteps = nsteps,
```

```

    is.stoch = FALSE
  )

```

---

 parse\_config

*Parse MetaRVM Configuration File*


---

### Description

Reads and parses a YAML configuration file for MetaRVM simulations, extracting all necessary parameters for epidemic modeling including population data, disease parameters, mixing matrices, vaccination schedules, and simulation settings.

### Usage

```
parse_config(config_file, return_object = FALSE)
```

### Arguments

config_file	Character string. Path to a YAML configuration file containing model parameters and settings.
return_object	Logical. If TRUE, returns a MetaRVMConfig object for method chaining and enhanced functionality. If FALSE (default), returns a named list for backward compatibility.

### Details

The function processes a YAML configuration file with the following main sections:

#### Simulation Configuration:

- random\_seed: Optional random seed for reproducibility in case of stochastic simulations or stochastic parameters
- nsim: Number of simulation instances (default: 1)
- start\_date: Simulation start date in MM/DD/YYYY format
- length: Simulation length in days
- checkpoint\_dir: Optional checkpoint directory for saving intermediate results
- checkpoint\_dates: Optional list of dates to save checkpoints.
- restore\_from: Optional path to restore simulation from checkpoint

#### Population Data:

- mapping: CSV file path containing population mapping with demographic categories age, race, zone. The file must contain columns population\_id, age, race, zone, where the population\_id is defined using natural numbers.

- **initialization:** CSV file with initial population states. The file must contains columns `population_id`, `N`, `S0`, `I0`, `V0`, `R0`.
- **vaccination:** CSV file with vaccination schedule over time. The first column must be dates in `MM/DD/YYYY` format. The rest of the columns must corresponds to respective subpopulations in the numeric order of `population_id`.

**Mixing Matrices:** Contact matrices for different time periods. Each CSV file must have a matrix of order  $(N_{\text{pop}} \times N_{\text{pop}})$ , where,  $N_{\text{pop}}$  is the number of subpopulations. It is assumed that the  $i$ -th row and  $j$ -th column correspond to  $i$ -th and  $j$ -th subpopulations.

- `weekday_day`, `weekday_night`: Weekday contact patterns
- `weekend_day`, `weekend_night`: Weekend contact patterns

**Disease Parameters:** Epidemiological parameters (can be scalars or distributions):

- `ts`: Transmission rate for symptomatic individuals
- `tv`: Transmission rate for vaccinated individuals
- `ve`: Vaccine effectiveness
- `de`, `dp`, `da`, `ds`, `dh`, `dr`: Duration parameters for different disease states
- `pea`, `psr`, `phr`: Probability parameters for disease transitions

**Sub-population Parameters:** `sub_disease_params` allows specification of different parameter values for specific demographic categories (e.g., age groups, races).

The function supports stochastic parameters through distribution specifications with `dist`, `mu`, `sd`, `shape`, `rate`, etc.

## Value

If `return_object = FALSE` (default), returns a named list containing:

**`N_pop`** Number of population groups

**`pop_map`** `Data.table` with population mapping and demographics

**`S_ini`, `E_ini`, `I_asymp_ini`, `I_presymp_ini`, `I_symp_ini`, `H_ini`, `D_ini`, `P_ini`, `V_ini`, `R_ini`** Initial compartment populations

**`vac_time_id`, `vac_counts`, `vac_mat`** Vaccination schedule data

**`m_wd_d`, `m_wd_n`, `m_we_d`, `m_we_n`** Contact mixing matrices

**`ts`, `tv`, `ve`, `dv`, `de`, `dp`, `da`, `ds`, `dh`, `dr`, `pea`, `psr`, `phr`** Disease parameter matrices ( $n_{\text{sim}} \times N_{\text{pop}}$ )

**`start_date`** Simulation start date as `Date` object

**`sim_length`** Simulation length in days

**`nsim`** Number of simulation instances

**`random_seed`** Random seed used (if any)

**`delta_t`** Time step size (fixed at 0.5)

**`chk_file_names`, `chk_time_steps`, `do_chk`** Checkpointing configuration

If `return_object = TRUE`, returns a `MetaRVMConfig` object with methods for parameter access and validation.

## Parameter Distributions

Disease parameters can be specified as distributions for stochastic modeling:

- **lognormal:** dist: "lognormal", mu: value, sd: value
- **gamma:** dist: "gamma", shape: value, rate: value
- **uniform:** dist: "uniform", min: value, max: value
- **beta:** dist: "beta", shape1: value, shape2: value
- **gaussian:** dist: "gaussian", mean: value, sd: value

## File Requirements

**Population mapping file** must contain columns:

- population\_id: Unique identifier for each population group, natural numbers
- age: Age category (e.g., "0-4", "5-11", "12-17", "18-49", "50-64", "65+")
- race: Race/ethnicity category
- zone: Geographic zone identifier

**Population initialization file** must contain: N (total population), S<sub>0</sub>, I<sub>0</sub>, V<sub>0</sub>, R<sub>0</sub> (initial compartment counts)

**Vaccination file** must contain: date (MM/DD/YYYY format) and vaccination counts for each population group

## Author(s)

Arindam Fadikar

## See Also

[metaRVM](#) for running simulations with parsed configuration [MetaRVMConfig](#) for the configuration object class [process\\_vac\\_data](#) for vaccination data processing

## Examples

```
options(odin.verbose = FALSE)
example_config <- system.file("extdata", "example_config.yaml", package = "MetaRVM")
# Parse configuration file and return list (backward compatible)
config <- parse_config(example_config)

# Parse and return MetaRVMConfig object for method chaining
config_obj <- parse_config(example_config, return_object = TRUE)

# Access parameters from config object
config_obj$get("N_pop")
config_obj$list_parameters()

# Use with MetaRVM simulation
results <- metaRVM(config_obj)
```

# Index

`format_metarvm_output`, [2](#), [19](#)  
`format_metarvm_output()`, [3](#)

`meta_sim`, [2](#), [14](#)  
`meta_sim()`, [3](#), [4](#)  
`MetaRVM`, [2](#)  
`metaRVM`, [3](#), [19](#), [22](#)  
`MetaRVM::MetaRVMConfig`, [5](#)  
`MetaRVMCheck`, [5](#)  
`MetaRVMConfig`, [3–5](#), [6](#), [22](#)  
`MetaRVMResults`, [3](#), [4](#), [9](#)  
`MetaRVMSummary`, [11](#)

`parse_config`, [19](#), [20](#)  
`parse_config()`, [3](#), [4](#)  
`process_vac_data`, [22](#)