User Manual: HERMES

2026-01-26

Contents

Introduction

This manual is intended for scientific users who wish to understand and apply HERMES, an individual-based, health- and economic risk-stratified microsimulation framework. The acronym HERMES stands for Health, Epidemic and Economic R-based Microsimulation Engine for individual-based Simulations. In its current configuration, the framework implements an individual-based infectious disease transmission model in a structured population comprising households, schools, workplaces, and the general community.

The model source code can be obtained by cloning the GitHub repository, or by downloading the individual R scripts as described in the Install section at the end of this manual.

The software is distributed in the hope that it will be useful, but without any warranty. For use in educational, research, or other scientific contexts, users are encouraged to contact the authors listed at the end of this document.

Getting started

Assuming the main file is present in the current directory, and the other model files in the `lib` directory, you can run the main workbench script as follows:

```
source("main.R")
```

The core API consists of `get_default_parameters()` and `run_ibm()`. The snippet below sources the model kernel and runs a default simulation.

```
source('lib/ibm_core.R')
params <- get_default_parameters()
out <- run_ibm(params)
```

The complete set of configurable parameters is returned as a named list. For documentation and reproducibility, it is often helpful to record parameter names and values alongside model outputs.

```
params <- get_default_parameters()
names(params)
print_model_parameters(params)
```

Model overview

The population is generated synthetically, with each individual represented as a row in a data frame. The final population size is fixed at `pop_size`. The default demography assumes nuclear households with adults (aged 19–80) and children (aged 1–18). Each individual is assigned: age, household ID, school classroom ID (if applicable), workplace ID (if applicable), and health and infection-related attributes.

Household formation

Households are generated initially assuming two adults and up to two children. The construction aims for realistic intergenerational age structures within households:

- Adult 1: age is sampled uniformly from ages 19-80.
- Adult 2: age is generated by adding a random age difference (-5 to +5 years) to Adult 1.
- Child 1: age is derived by subtracting a household age gap (22-35 years) from the youngest adult.
- Child 2: age is derived by subtracting a sibling age gap (1 - 4 years) from Child 1.

After sampling, children with invalid ages (below 1 or above 19) are removed. If the resulting population exceeds `pop_size`, individuals are randomly removed, which can also yield incomplete households.

School assignment

School assignment is age-based. Target ages for school enrollment are controlled by `target_school_ages` (default 3–18 years). The number of schools is derived from the target school size (`target_school_size`) and the number of eligible children in the population. Each eligible individual is assigned to a classroom labeled by age and school index. Individuals outside the target ages have `classroom_id = NA`.

Workplace assignment

Workplace assignment is applied to adults. Eligible ages are defined by `target_workplace_ages` (default 19–65 years). The number of workplaces is derived from `target_workplace_size` and the number of eligible adults. Workplace IDs are sampled for each eligible adult; other individuals have `workplace_id = NA`. Random sampling can yield workplaces that are smaller or larger than the target size.

Health states and individual attributes

Each individual has a categorical health state:

- **S**: Susceptible
- **I**: Infected and infectious
- **R**: Recovered
- **V**: Vaccinated
- **D**: Dead

Additional tracked variables include: `infector`, `infector_age`, `time_of_infection`, `generation_interval`, and `secondary_cases`. Vaccination is implemented by assigning a fraction `vaccine_coverage` of the population to state **V** at initialization; vaccinated individuals experience reduced infection probability proportional to `vaccine_effectiveness`.

Transmission dynamics

Transmission is simulated explicitly for each infected individual on each day. For every infectious individual, potential transmission is evaluated in four settings: household, community, school, and workplace. Each setting has a setting-specific effective contact probability. The community contact probability applies to all susceptible individuals; household, school, and workplace contributions are added for individuals sharing the corresponding location with the infector. Setting-specific contact probabilities are multiplied by a per-contact transmission probability, and infection events are sampled using Bernoulli trials. All transmission events are logged with infector identity and age, time of infection, and generation interval. For each infector, the number of secondary cases is tracked.

Disease natural history

Recovered individuals transition from infected to recovered after an average duration of `num_days_infected` days. The transition rate is converted to a daily recovery probability assuming an exponential process. Mortality includes two age-specific components: general mortality and disease-induced mortality (applied only to infected individuals). If death and recovery occur on the same day, death overrides recovery.

Simulation loop and outputs

At simulation start, `num_infected_seeds` individuals are randomly selected and set to infected; their time of infection is set to day 0. The model runs for `num_days` daily time steps. At each day: new infections, recoveries, and deaths are sampled, and health states are logged. Outputs include:

(i) health state proportions over time;
(ii) the parameter set used;
(iii) the individual-level population data structure at the final time step.

Default plots include: time series of **S/I/R/V/D**, secondary case distributions, generation interval distributions, and an age-to-age transmission matrix (3-year age bins).

Parameter summary

Key parameters include: `pop_size`, `num_days`, `num_infected_seeds`, `vaccine_coverage`, `num_days_infected`, `transmission_prob`, `num_contacts_community_day`, `contact_prob_household`, `contact_prob_school`, `contact_prob_workplace`, `target_school_size`, `target_workplace_size`, `rng_seed`, and age-specific general and disease-related mortality rates.

Model output

Simulation output is written to the `output` directory, in a subdirectory whose name is specified via the `params` object. By default, output is stored in `output/ibm_flu`

Results are saved as RDS files and include:

- **`health_states.rds`**
  A list containing:

  - `log_health`: a data frame with population-level health state proportions over time (columns represent health states; rows represent time steps),
  - `params`: the full set of model parameters used for the simulation.
- **`pop_matrix.rds`**
  A data frame containing individual-level population data at the end of the simulation, with one row per individual and individual attributes stored in columns.

Reproducible example run

The following chunk illustrates a minimal run that is suitable for inclusion in reports. It prints the available parameter names, modifies a small subset, and runs the model.

```
# Source the model kernel (adjust path to match your repository layout)
source('./lib/ibm_core.R')

# Load default parameters
params <- get_default_parameters()

# Display parameter names (useful for documentation and sensitivity analyses)
names(params)

# Modify a small number of parameters for demonstration
params$num_infected_seeds <- 10
params$bool_add_baseline  <- TRUE

# Run the model
out <- run_ibm(params, verbose = TRUE)

# Inspect outputs
str(out)
head(out$log_health)

# Inspect final population details
pop_data_file <- file.path(out$params$output_dir,'pop_data.rds')
pop_data <- readRDS(pop_data_file)
dim(pop_data)
head(pop_data)
```

Regression testing

HERMES provides an internal regression test to detect unintended changes to default behaviour.

```
run_ibm_regression_test()
```

Authors

Lander Willem
University of Antwerp, Belgium (lander.willem@uantwerpen.be)

et al.

Appendix: Install

All R files can be obtained using `download.file()` as explained in our install.R file or in the code below:

```
# ------------------------------------------------------------------------- -
# USAGE NOTES:
# - Set your working directory to the desired project root before running
#   this script.
# - All helper files will be downloaded into a local 'lib/' directory.
# - Existing files in 'lib/' with the same name will be overwritten.
# ------------------------------------------------------------------------- -

# Required dependency for interacting with the GitHub API
library(jsonlite)

# GitHub API endpoint pointing to the 'lib' directory of the HERMES repository
github_api_url <- "https://api.github.com/repos/lwillem/ibm_hermes/contents/lib"

# Local destination directory for helper scripts
project_dir <- "ibm_hermes"

# Query the GitHub API to retrieve metadata for files in the repository
repo_contents <- fromJSON(github_api_url)

# PREPARE LOCAL DIRECTORY
# Create local project directory if it does not yet exist
dir.create(project_dir, showWarnings = FALSE, recursive = TRUE)

# DOWNLOAD HELPER FILES
for (i in seq_len(nrow(repo_contents))) {
  download.file(
    url  = repo_contents$download_url[i],
    destfile = file.path(project_dir, 'lib', repo_contents$name[i]),
    mode = "wb"
  )
}

# DOWNLOAD MAIN and DESCRIPTION FILES
repo_meta_content <- fromJSON(dirname(github_api_url))

download.file(
  url = repo_meta_content$download_url[repo_meta_content$name == 'main.R'],
  destfile = file.path(dirname(project_dir), 'main.R'),
  mode = "wb",
)

download.file(
  url = repo_meta_content$download_url[repo_meta_content$name == 'DESCRIPTION.txt'],
  destfile = file.path(project_dir, 'DESCRIPTION.txt'),
  mode = "wb",
)
```