Package: epicR (via r-universe)

May 10, 2026

Title Evaluation Platform in Chronic Obstructive Pulmonary Disease

Version 1.0.1

Author Mohsen Sadatsafavi [aut, cph], Amin Adibi [aut, cre], Kate Johnson [aut]

Maintainer Amin Adibi <adibi@alumni.ubc.ca>

Description Evaluation Platform in Chronic Obstructive Pulmonary Disease (EPIC) is a Discrete Event Simulation (DES) model that simulates health outcomes of patients with Chronic Obstructive Pulmonary Disease (COPD) based on demographics and individual-level risk factors, based on the model published in Sadatsafavi et al. (2019) <doi:10.1177/0272989X18824098>.

Depends R (>= 4.1.0)

License GPL-3

Encoding UTF-8

Imports Rcpp (>= 1.0.0), graphics, stats, tools, ggplot2 (>= 4.0.0), ggthemes, sqldf, jsonlite, readr, reshape2, scales, tidyr, tibble, dplyr

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0), openxlsx, survival, survminer, knitr, rmarkdown

Config/testthat/edition 3

Config/build/no-staged-install TRUE

VignetteBuilder knitr, rmarkdown

Config/pak/sysreqs libicu-dev libx11-dev

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

Date/Publication 2026-03-11 17:36:36 UTC

RemoteUrl https://github.com/resplab/epicr

RemoteRef HEAD

RemoteSha 5c8938c7c76cb5ee98fcd6906272d03a4648c941

Contents

`epicR-package` `epicR` *package*

Description

Evaluation Platform in COPD (EPIC) R Package

Details

See the README on GitHub

Author(s)

Maintainer: Amin Adibi <adibi@alumni.ubc.ca>

Authors:

- Mohsen Sadatsafavi <mohsen.sadatsafavi@ubc.ca> [copyright holder]
- Kate Johnson <kate.johnson@alumni.ubc.ca>

calc_event_stack_size *Calculate recommended event_stack_size for a given number of agents*

Description

Calculate recommended event_stack_size for a given number of agents

Usage

```
calc_event_stack_size(n_agents, time_horizon = 20)
```

Arguments

n_agents	number of agents to simulate
time_horizon	simulation time horizon in years (default 20)

Value

recommended event_stack_size

calibrate_COPD_inc *Solves stochastically for COPD incidence rate equation.*

Description

Solves stochastically for COPD incidence rate equation.

Usage

```
calibrate_COPD_inc(
  nIterations = 100,
  nPatients = 1e+05,
  time_horizon = 20,
  output_dir = tempdir()
)
```

Arguments

nIterations	number of iterations for the numerical solution
nPatients	number of simulated agents.
time_horizon	in years
output_dir	directory for output CSV files (default: tempdir())

Value

regression co-efficients as files

`calibrate_explicit_mortality` *Calibrates explicit mortality*

Description

Calibrates explicit mortality

Usage

```
calibrate_explicit_mortality(n_sim = 10^8)
```

Arguments

`n_sim`	number of agents

Value

difference in mortality rates and life table

`calibrate_explicit_mortality2` *Calibrates explicit mortality by Amin*

Description

Calibrates explicit mortality by Amin

Usage

```
calibrate_explicit_mortality2(n_sim = 10^7)
```

Arguments

`n_sim`	number of agents

Value

difference in mortality rates and life table

calibrate_smoking *Calibrates smoking*

Description

Calibrates smoking

Usage

```
calibrate_smoking()
```

Value

TODO

copy_configs_to_user *Copy default configs to user directory*

Description

Copy default configs to user directory

Usage

```
copy_configs_to_user(overwrite = FALSE)
```

Arguments

overwrite	Whether to overwrite existing user configs (default FALSE)

Value

TRUE if successful, FALSE otherwise

epicR-config *epicR Configuration Management*

Description

The epicR package now supports user-customizable configuration files. When the package is loaded for the first time, configuration files are automatically copied to a user-specific directory following CRAN policy. You can modify these files to customize model parameters for your specific region or research needs.

Configuration Directory

User configuration files are stored in a platform-specific location determined by `tools::R_user_dir("epicR", "config")`. Use `get_user_config_dir()` to find the exact path on your system.

Available Functions

- `get_user_config_dir`: Get the path to user config directory
- `user_config_exists`: Check if a user config exists
- `copy_configs_to_user`: Copy default configs to user directory
- `reset_user_configs`: Reset configs to package defaults
- `list_available_jurisdictions`: List available config jurisdictions
- `validate_config`: Validate a configuration file

Usage

The package automatically uses user configs if they exist. When you call `get_input()`, it will:

1. First check for user config files in the user config directory
2. If found, use the user's customized configuration
3. If not found, fall back to package default configurations

Customizing Configurations

To customize configurations for your region:

1. The config files are automatically copied on first package load
2. Edit the JSON files (e.g., 'config_canada.json', 'config_us.json')
3. Save your changes
4. The next time you use `get_input()`, your changes will be used

Adding New Jurisdictions

To add a new jurisdiction:

1. Copy an existing config file (e.g., ‘config_canada.json‘)
2. Rename it to ‘config_yourcountry.json‘
3. Update the "jurisdiction" field in the JSON to match
4. Modify all parameters as needed for your region
5. Use with: `get_input(jurisdiction = "yourcountry")`

Resetting to Defaults

If you need to reset your configurations:

- Reset all configs: `reset_user_configs()`
- Reset specific jurisdiction: `reset_user_configs("canada")`

Examples

```
# Check where user configs are stored
get_user_config_dir()

# List available jurisdictions
list_available_jurisdictions()

# Validate your custom config
validate_config("canada", user = TRUE)

# Reset to package defaults if needed
reset_user_configs("canada")

# Use your custom config
input <- get_input(jurisdiction = "canada")
```

`estimate_memory_required` *Estimate memory required for simulation*

Description

Estimate memory required for simulation

Usage

```
estimate_memory_required(n_agents, record_mode = 0, time_horizon = 20)
```

Arguments

n_agents	number of agents
record_mode	recording mode (0=none, 1=agent, 2=event, 3=some_event)
time_horizon	simulation time horizon in years

Value

estimated memory in bytes

---

export_figures *Runs the model and exports an excel file with all output data*

Description

Runs the model and exports an excel file with all output data

Usage

```
export_figures(nPatients = 10000)
```

Arguments

nPatients	number of agents

Value

an excel file with all output

---

express_matrix *Express matrix.*

Description

Takes a named matrix and writes the R code to populate it; useful for generating input expressions from calibration results.

Usage

```
express_matrix(mtx)
```

Arguments

mtx	a matrix

Value

Invisibly returns the generated R code as a character string, also outputs via message().

get_agent_events *Returns all events of an agent.*

Description

Returns all events of an agent.

Returns events specific to an agent.

Usage

```
get_agent_events(id)

get_agent_events(id)
```

Arguments

id	Agent number

Value

all events of agent id

dataframe consisting all events specific to agent id

get_agent_size_bytes *Returns the size of agent struct in bytes*

Description

Returns the size of agent struct in bytes

Get size of agent struct in bytes (from C code)

Usage

```
get_agent_size_bytes()

get_agent_size_bytes()
```

Value

size of agent struct in bytes

size in bytes

get_all_events *Returns all events.*

Description

Returns all events.

Returns all events.

Usage

```
get_all_events()
```

```
get_all_events()
```

Value

all events

dataframe consisting all events.

get_all_events_matrix *Returns a matrix containing all events*

Description

Returns a matrix containing all events

Usage

```
get_all_events_matrix()
```

Value

a matrix containing all events

get_available_memory *Get available system memory (platform-specific)*

Description

Get available system memory (platform-specific)

Usage

```
get_available_memory()
```

Value

available memory in bytes

get_default_settings *Exports default settings*

Description

Exports default settings

Usage

```
get_default_settings()
```

Value

default settings

get_errors *Returns errors*

Description

Returns errors

Usage

```
get_errors()
```

Value

a text with description of error messages

get_event *Returns the events stack.*

Description

Returns the events stack.

Usage

```
get_event(i)
```

Arguments

i	number of event

Value

events

get_events_by_type *Returns all events of a certain type.*

Description

Returns all events of a certain type.

Returns certain events by type

Usage

```
get_events_by_type(event_type)

get_events_by_type(event_type)
```

Arguments

event_type	event_type number

Value

all events of the type `event_type`

dataframe consisting all events of the type `event_type`

`get_input` *Returns a list of default model input values*

Description

Returns a list of default model input values

Usage

```
get_input(
  age0 = 40,
  time_horizon = 20,
  discount_cost = 0,
  discount_qaly = 0.03,
  closed_cohort = 0,
  jurisdiction = "canada"
)
```

Arguments

`age0`	Starting age in the model
`time_horizon`	Model time horizon
`discount_cost`	Discounting for cost outcomes
`discount_qaly`	Discounting for QALY outcomes
`closed_cohort`	Whether the model should run as closed_cohort, open-population by default.
`jurisdiction`	Jurisdiction for model parameters ("canada" or "us")

Value

A list with four components:

values A nested list of model input parameter values

help A nested list of help text descriptions for each parameter

ref A nested list of reference information for each parameter

config The raw configuration object loaded from the JSON file

get_inputs *Returns inputs*

Description

Returns inputs

Usage

```
get_inputs()
```

Value

all inputs

get_list_elements *Get list elements*

Description

Recursively extracts element names from a nested list structure.

Usage

```
get_list_elements(ls, running_name = "")
```

Arguments

ls	A list to extract element names from
running_name	Internal parameter for recursion (default: "")

Value

A character vector of element names, with nested elements separated by "$" (e.g., "parent$child$grandchild").

get_n_events *Returns total number of events.*

Description

Returns total number of events.

Usage

```
get_n_events()
```

Value

number of events

get_output *Main outputs of the current run.*

Description

Main outputs of the current run.

Usage

```
get_output()
```

Value

number of agents, cumulative time, number of deaths, number of COPD cases, as well as exacerbation statistics and QALYs.

get_output_ex *Extra outputs from the model*

Description

Extra outputs from the model

Usage

```
get_output_ex()
```

Value

Extra outputs from the model.

get_progress *Returns current simulation progress.*

Description

Returns current simulation progress.

Usage

```
get_progress()
```

Value

Number of agents processed so far (last_id).

get_runtime_stats *Returns run time stats.*

Description

Returns run time stats.

Usage

```
get_runtime_stats()
```

Value

agent size as well as memory and random variable fill stats.

get_sample_output *Returns a sample output for a given year and gender.*

Description

Returns a sample output for a given year and gender.

Usage

```
get_sample_output(year, sex)
```

Arguments

year	a number
sex	a number, 0 for male and 1 for female

Value

that specific output

`get_settings` *Returns current settings.*

Description

Returns current settings.

Usage

```
get_settings()
```

Value

current settings.

`get_smith` *Returns agent Smith.*

Description

Returns agent Smith.

Usage

```
get_smith()
```

Value

agent smith.

`get_user_config_dir` *Get user config directory path*

Description

Returns the path to the user configuration directory following CRAN policy using `tools::R_user_dir()`.

Usage

```
get_user_config_dir()
```

Value

Path to user config directory

`init_session` *Initializes a model. Allocates memory to the C engine.*

Description

Initializes a model. Allocates memory to the C engine.

Usage

```
init_session(settings = get_default_settings(), jurisdiction = "canada")
```

Arguments

`settings`	customized settings.
`jurisdiction`	The jurisdiction for which to load input parameters (default: "canada").

Value

0 if successful.

`list_available_jurisdictions` *List available config jurisdictions*

Description

List available config jurisdictions

Usage

```
list_available_jurisdictions()
```

Value

Character vector of available jurisdictions

mvrnormArma *Samples from a multivariate normal*

Description

Samples from a multivariate normal

Usage

```
mvrnormArma(n, mu, sigma)
```

Arguments

n	number of samples to be taken
mu	the mean
sigma	the covariance matrix

Value

the multivariate normal sample

report_COPD_by_ctime *Reports COPD related stats.*

Description

Reports COPD related stats.

Usage

```
report_COPD_by_ctime(n_sim = 10^6)
```

Arguments

n_sim	number of simulated agents.

Value

COPD-related stats

report_exacerbation_by_time *Reports exacerbation-related stats.*

Description

Reports exacerbation-related stats.

Usage

```
report_exacerbation_by_time(n_sim = 10^5)
```

Arguments

n_sim	number of simulated agents.

Value

exacerbation-related stats

reset_model_input *Reset the cached model input*

Description

Forces the default model_input to be reloaded on next access. Useful after modifying config files.

Usage

```
reset_model_input()
```

Value

No return value, called for side effects (clears the model input cache).

reset_user_configs *Reset user configs to package defaults*

Description

Reset user configs to package defaults

Usage

```
reset_user_configs(jurisdiction = NULL)
```

Arguments

jurisdiction	Specific jurisdiction to reset, or NULL for all (default NULL)

Value

TRUE if successful, FALSE otherwise

resume *Resumes running of model.*

Description

Resumes running of model.

Usage

```
resume(max_n_agents = NULL)
```

Arguments

max_n_agents	maximum number of agents

Value

0 if successful.

sanity_check *Basic tests of model functionality. Serious issues if the test does not pass.*

Description

Basic tests of model functionality. Serious issues if the test does not pass.

Usage

```
sanity_check()
```

Value

tests results

sanity_COPD *Basic COPD test.*

Description

Basic COPD test.

Usage

```
sanity_COPD()
```

Value

validation test results

set_input_var *Sets input variables.*

Description

Sets input variables.

Usage

```
set_input_var(name, value)
```

Arguments

name	a string
value	a number

Value

0 if successful

---

set_settings_var *Sets model settings.*

Description

Sets model settings.

Usage

```
set_settings_var(name, value)
```

Arguments

name	a name
value	a value

Value

0 if successful.

---

simulate *Convenience function: run simulation and return results*

Description

This is a simplified interface that handles session management automatically and returns the results directly. Ideal for most users. Progress information is displayed including: configuration summary, a real-time progress bar showing percentage completion (10

Usage

```
simulate(
  input = NULL,
  settings = NULL,
  jurisdiction = "canada",
  time_horizon = NULL,
  n_agents = NULL,
  extended_results = TRUE,
  return_events = FALSE,
  seed = NULL
)
```

Arguments

input	customized input criteria (optional)
settings	customized settings (optional)
jurisdiction	Jurisdiction for model parameters ("canada" or "us")
time_horizon	Model time horizon in years (default: uses config file value)
n_agents	Number of agents to simulate (default: 60,000)
extended_results	whether to return extended results in addition to basic (default: TRUE)
return_events	whether to return event matrix (default: FALSE). If TRUE, automatically sets record_mode=2
seed	Random seed for reproducibility (optional). If provided, ensures identical results across runs

Value

list with simulation results (always includes 'basic', includes 'extended' by default, optionally 'events')

Examples

```
# Simplest usage - includes both basic and extended results
results <- simulate()
print(results$basic)
print(results$extended)

# With custom parameters
results <- simulate(jurisdiction = "us", time_horizon = 10, n_agents = 100000)

# Quick test with fewer agents
results <- simulate(n_agents = 10000)

# Basic output only (faster, less memory)
results <- simulate(extended_results = FALSE)
print(results$basic)
```

```
# With event history
results <- simulate(return_events = TRUE)
print(results$events)  # Event matrix

# With reproducible results
results1 <- simulate(seed = 123)
results2 <- simulate(seed = 123)
# results1 and results2 will be identical
```

`terminate_session` *Terminates a session and releases allocated memory.*

Description

Terminates a session and releases allocated memory.

Usage

```
terminate_session()
```

Value

0 if successful.

`test_case_detection` *Returns results of Case Detection strategies*

Description

Returns results of Case Detection strategies

Usage

```
test_case_detection(
  n_sim = 10000,
  p_of_CD = 0.1,
  min_age = 40,
  min_pack_years = 0,
  only_smokers = 0,
  CD_method = "CDQ195"
)
```

Arguments

n_sim	number of agents
p_of_CD	probability of recieving case detection given that an agent meets the selection criteria
min_age	minimum age that can recieve case detection
min_pack_years	minimum pack years that can recieve case detection
only_smokers	set to 1 if only smokers should recieve case detection
CD_method	Choose one case detection method: CDQ195", "CDQ165", "FlowMeter", "FlowMeter_CDQ"

Value

results of case detection strategy compared to no case detection

---

user_config_exists *Check if user config exists*

Description

Check if user config exists

Usage

```
user_config_exists(jurisdiction = "canada")
```

Arguments

jurisdiction	Jurisdiction name (e.g., "canada", "us")

Value

TRUE if user config exists, FALSE otherwise

validate_config *Validate a config file*

Description

Validate a config file

Usage

```
validate_config(jurisdiction = "canada", user = TRUE)
```

Arguments

jurisdiction	Jurisdiction to validate
user	Whether to validate user config (TRUE) or package config (FALSE)

Value

TRUE if valid, FALSE otherwise (with warnings about issues)

validate_COPD *Returns results of validation tests for COPD*

Description

This function runs validation tests for COPD model outputs. It estimates the baseline prevalence and incidence of COPD, along with sex-specific baseline COPD prevalence. Additionally, it calculates the baseline prevalence of COPD by age groups and smoking pack-years. It also estimates the coefficients for the relationships between age, pack-years, smoking status, and the prevalence of COPD.

Usage

```
validate_COPD(incident_COPD_k = 1, return_CI = FALSE, jurisdiction = "canada")
```

Arguments

incident_COPD_k	a number (default=1) by which the incidence rate of COPD will be multiplied.
return_CI	if TRUE, returns 95 percent confidence intervals for the "Year" coefficient
jurisdiction	character string specifying the jurisdiction for validation ("canada" or "us"). Default is "canada".

Value

For Canada: list with validation test results. For US: data frame with COPD prevalence by age group over time.

validate_diagnosis *Returns results of validation tests for diagnosis*

Description

This function returns a table showing the proportion of COPD patients diagnosed over the model's runtime. It also provides a second table that breaks down the proportion of diagnosed patients by COPD severity. Additionally, the function generates a plot to visualize the distribution of diagnoses over time.

Usage

```
validate_diagnosis(n_sim = 10000, jurisdiction = "canada")
```

Arguments

n_sim	number of agents
jurisdiction	character string specifying the jurisdiction ("canada" or "us"). Default is "canada"

Value

validation test results

validate_exacerbation *Returns results of validation tests for exacerbation rates*

Description

This function runs validation tests for COPD exacerbation rates by GOLD stage and compares them with reference values from studies such as CanCOLD, CIHI, and Hoogendoorn. It simulates exacerbation events, and returns key metrics, including overall exacerbation rates, rates by GOLD stage, and rates in diagnosed vs. undiagnosed patients.

Usage

```
validate_exacerbation(
  base_agents = 10000,
  input = NULL,
  jurisdiction = "canada"
)
```

Arguments

base_agents	Number of agents in the simulation. Default is 1e4.
input	EPIC inputs
jurisdiction	character string specifying the jurisdiction for validation ("canada" or "us"). Default is "canada".

Value

For Canada: validation test results (invisible). For US: invisible NULL (results displayed via messages and plots).

---

validate_gpvisits *Returns results of validation tests for GP visits*

Description

This function returns Average number of GP visits by sex, COPD severity and COPD diagnosis status along with their plots.

Usage

```
validate_gpvisits(n_sim = 10000, jurisdiction = "canada")
```

Arguments

n_sim	number of agents
jurisdiction	character string specifying the jurisdiction ("canada" or "us"). Default is "canada"

Value

validation test results

---

validate_lung_function *Returns results of validation tests for lung function*

Description

This function evaluates FEV1 (Forced Expiratory Volume in 1 second) values and GOLD stage distributions to assess lung function in simulated patients.

Usage

```
validate_lung_function(jurisdiction = "canada")
```

Arguments

jurisdiction	character string specifying the jurisdiction ("canada" or "us"). Default is "canada"

Value

validation test results

validate_medication *Returns results of validation tests for medication module.*

Description

This function returns plots showing medication usage over time

Usage

```
validate_medication(n_sim = 50000, jurisdiction = "canada")
```

Arguments

n_sim	number of agents
jurisdiction	character string specifying the jurisdiction ("canada" or "us"). Default is "canada"

Value

validation test results for medication

validate_mortality *Returns results of validation tests for mortality rate*

Description

This function returns a table and a plot of the difference between simulated and expected (life table) mortality, by sex and age.

Usage

```
validate_mortality(
  n_sim = 5e+05,
  bgd = 1,
  bgd_h = 1,
  manual = 1,
  exacerbation = 1,
  jurisdiction = "canada"
)
```

Arguments

n_sim	number of simulated agents
bgd	a number
bgd_h	a number
manual	a number
exacerbation	a number
jurisdiction	character string specifying the jurisdiction ("canada" or "us"). Default is "canada"

Value

validation test results

`validate_overdiagnosis` *Returns results of validation tests for overdiagnosis*

Description

This function returns the proportion of non-COPD subjects overdiagnosed over model time.

Usage

```
validate_overdiagnosis(n_sim = 10000, jurisdiction = "canada")
```

Arguments

`n_sim`	number of agents
`jurisdiction`	character string specifying the jurisdiction ("canada" or "us"). Default is "canada"

Value

validation test results

`validate_payoffs` *Returns results of validation tests for payoffs, costs and QALYs*

Description

Returns results of validation tests for payoffs, costs and QALYs

Usage

```
validate_payoffs(
  nPatient = 1e+06,
  disableDiscounting = TRUE,
  disableExacMortality = TRUE,
  jurisdiction = "canada"
)
```

Arguments

nPatient	number of simulated patients. Default is 1e6.
disableDiscounting	if TRUE, discounting will be disabled for cost and QALY calculations. Default: TRUE
disableExacMortality	if TRUE, mortality due to exacerbations will be disabled for cost and QALY calculations. Default: TRUE
jurisdiction	character string specifying the jurisdiction for validation ("canada" or "us"). Default is "canada". Currently only "canada" is implemented.

Value

validation test results

---

validate_population *Returns simulated vs. predicted population table and a plot*

Description

Returns simulated vs. predicted population table and a plot

Usage

```
validate_population(
  remove_COPD = 0,
  incidence_k = 1,
  savePlots = 0,
  jurisdiction = "canada"
)
```

Arguments

remove_COPD	0 or 1, indicating whether COPD-caused mortality should be removed
incidence_k	a number (default=1) by which the incidence rate of population will be multiplied.
savePlots	0 or 1, exports 300 DPI population growth and pyramid plots comparing simulated vs. predicted population
jurisdiction	character string specifying the jurisdiction for validation ("canada" or "us"). Default is "canada".

Value

For Canada: returns a table showing predicted (StatsCan) and simulated population values. For US: returns a data frame with population comparisons by age group and year.

validate_smoking *Returns results of validation tests for smoking module.*

Description

It compares simulated smoking prevalence and trends against observed data to assess the model's accuracy.

Usage

```
validate_smoking(remove_COPD = 1, intercept_k = NULL, jurisdiction = "canada")
```

Arguments

remove_COPD	0 or 1. whether to remove COPD-related mortality.
intercept_k	a number
jurisdiction	character string specifying the jurisdiction for validation ("canada" or "us"). Default is "canada".

Value

For Canada: validation test results (invisible). For US: data frame with smoking status rates by year.

validate_survival *Returns the Kaplan Meier curve comparing COPD and non-COPD*

Description

Returns the Kaplan Meier curve comparing COPD and non-COPD

Usage

```
validate_survival(
  savePlots = FALSE,
  base_agents = 10000,
  jurisdiction = "canada"
)
```

Arguments

savePlots	TRUE or FALSE (default), exports 300 DPI population growth and pyramid plots comparing simulated vs. predicted population
base_agents	Number of agents in the simulation. Default is 1e4.
jurisdiction	character string specifying the jurisdiction ("canada" or "us"). Default is "canada"

Value

validation test results

`validate_symptoms` *Returns results of validation tests for Symptoms*

Description

This function plots the prevalence of cough, dyspnea, phlegm and wheeze over time and by GOLD stage.

Usage

```
validate_symptoms(n_sim = 10000, jurisdiction = "canada")
```

Arguments

`n_sim`	number of agents
`jurisdiction`	character string specifying the jurisdiction ("canada" or "us"). Default is "canada"

Value

validation test results

`validate_treatment` *Returns results of validation tests for Treatment*

Description

This function runs validation tests to examine how treatment initiated at diagnosis influences exacerbation rates in COPD patients. It compares exacerbation rates between diagnosed and undiagnosed patients and assesses the impact of treatment.

Usage

```
validate_treatment(n_sim = 10000, jurisdiction = "canada")
```

Arguments

`n_sim`	number of agents
`jurisdiction`	character string specifying the jurisdiction ("canada" or "us"). Default is "canada"

Value

validation test results

Index