

Package: mergeGridR (via r-universe)

June 25, 2026

Title Grid-Based Number Merge Puzzle Simulation

Date 2026-06-10

Version 0.2.0

Author Frederic Bertrand [cre, aut]
(<https://orcid.org/0000-0002-0837-8281>)

Maintainer Frederic Bertrand <frederic.bertrand@lecnam.net>

Description Provides tools to simulate, analyse, visualise, and benchmark grid-based number merge puzzles. The package implements generic grid mechanics, tile-spawning rules, merge rules, scoring functions, reproducible simulation utilities, and local 'Shiny' and 'WebGL' interfaces for interactive use. It is intended for teaching, algorithmic experimentation, and game-theoretic examples. The autoplay helpers use standard heuristic search and Monte Carlo simulation ideas described in Russell and Norvig (2021, ISBN:9780134610993) and Robert and Casella (2004, ISBN:9780387212395).

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports ggplot2, ggWebGL, parallel, Rcpp, shiny, tools

LinkingTo Rcpp

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

URL <https://github.com/fbertran/mergeGridR/>,
<https://fbertran.github.io/mergeGridR/>

BugReports <https://github.com/fbertran/mergeGridR/issues>

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

Repository <https://fbertran.r-universe.dev>

Date/Publication 2026-06-13 12:14:03 UTC

RemoteUrl <https://github.com/fbertran/mergegridr>

RemoteRef HEAD

RemoteSha 8911b7c7fbbbc9e9ef740e8f864f10823115c30c

Contents

mergeGridR-package	2
autoplay_benchmark_results	3
autoplay_benchmark_settings	3
autoplay_game	4
autoplay_move	5
benchmark_autoplay_strategies	6
continue_game	7
drop_tile	7
export_static_app	8
game_config	8
get_high_score	9
new_game	10
reset_high_score	10
run_drop_number	11
Index	12

mergeGridR-package	<i>mergeGridR package</i>
--------------------	---------------------------

Description

Provides tools to simulate, analyse, visualise, and benchmark grid-based number merge puzzles. The package implements generic grid mechanics, tile-spawning rules, merge rules, scoring functions, reproducible simulation utilities, and local 'Shiny' and 'WebGL' interfaces for interactive use. It is intended for teaching, algorithmic experimentation, and game-theoretic examples. The autoplay helpers use standard heuristic search and Monte Carlo simulation ideas described in Russell and Norvig (2021, ISBN:9780134610993) and Robert and Casella (2004, ISBN:9780387212395).

Author(s)

Maintainer: Frederic Bertrand <frederic.bertrand@lecnam.net> ([ORCID](#))

See Also

Useful links:

- <https://github.com/fbertran/mergeGridR/>
- <https://fbertran.github.io/mergeGridR/>
- Report bugs at <https://github.com/fbertran/mergeGridR/issues>

autoplay_benchmark_results

Example autoplay benchmark results

Description

A small deterministic benchmark produced by `benchmark_autoplay_strategies()` using `autoplay_benchmark_settings` with `n_games = 5`, `max_moves = 50`, and `seed = 20260609`.

Usage

```
autoplay_benchmark_results
```

Format

A list with five elements:

summary Aggregate metrics by autoplay setting.

runs One row per simulated game and setting.

settings The benchmark settings grid.

config The game configuration.

seed The top-level benchmark seed.

autoplay_benchmark_settings

Autoplay benchmark setting presets

Description

Autoplay benchmark setting presets

Usage

```
autoplay_benchmark_settings(preset = c("quality", "fast"))
```

Arguments

preset Preset grid to return. "quality" compares a broader set of stronger settings; "fast" keeps the grid small for quick checks.

Value

A data frame with columns setting_id, strategy, depth, beam_width, simulations, and horizon.

autoplay_game	<i>Play a game with autoplay</i>
---------------	----------------------------------

Description

Play a game with autoplay

Usage

```
autoplay_game(
  state = new_game(),
  strategy = "lookahead",
  max_moves = 1000L,
  keep_states = FALSE,
  ...
)
```

Arguments

state Initial game state.

strategy Strategy passed to [autoplay_move\(\)](#).

max_moves Maximum number of moves to play.

keep_states If TRUE, include each intermediate state in the result.

... Additional arguments passed to [autoplay_move\(\)](#).

Value

A list with final_state, history, and states.

 autoplay_move

Choose an autoplay move

Description

autoplay_move() scores the legal columns for a game state and returns the recommended column without changing the state. The default strategy is a depth-limited lookahead search. future_mode = "visible" plays fairly from the visible queue and uses independent planning seeds for unknown future tiles; future_mode = "rng" allows simulations to use the state's internal deterministic random-number stream.

Usage

```
autoplay_move(
  state,
  strategy = c("lookahead", "growth_lookahead", "monte_carlo", "greedy"),
  depth = 3L,
  simulations = 100L,
  horizon = 30L,
  beam_width = 10L,
  future_mode = c("visible", "rng"),
  seed = NULL
)
```

Arguments

state	A game-state list returned by <code>new_game()</code> or <code>drop_tile()</code> .
strategy	Move-selection strategy: "lookahead", "growth_lookahead", "monte_carlo", or "greedy".
depth	Lookahead depth for the "lookahead" and "growth_lookahead" strategies.
simulations	Number of rollout simulations for "monte_carlo".
horizon	Maximum rollout length for "monte_carlo".
beam_width	Number of states retained per lookahead layer.
future_mode	Whether planning uses only visible future information or the state's deterministic RNG stream.
seed	Optional positive integer seed for visible-future planning.

Value

A list with column, strategy, future_mode, score_estimate, and a candidates data frame.

Reinforcement-learning extensions

Natural next steps include tabular Q-learning on handcrafted board features, approximate Q-learning using the current heuristic features, Deep Q-Networks with legal-action masks, PPO or actor-critic policy gradients, MCTS/UCT rollouts over the deterministic engine, and evolution strategies for tuning heuristic weights before introducing neural models.

```
benchmark_autoplay_strategies
      Benchmark autoplay strategies
```

Description

Runs repeated seeded games for each autoplay strategy/settings row and returns both per-game runs and aggregate summaries. Parallel execution uses base R PSOCK clusters when workers > 1; sequential execution is the default for reproducible package checks.

Usage

```
benchmark_autoplay_strategies(
  config = game_config(),
  n_games = 50L,
  max_moves = 1000L,
  settings = autoplay_benchmark_settings("quality"),
  future_mode = c("visible", "rng"),
  seed = NULL,
  workers = 1L
)
```

Arguments

config	A configuration list created by <code>game_config()</code> .
n_games	Number of games to run for each setting.
max_moves	Maximum moves per game.
settings	Strategy/settings grid from <code>autoplay_benchmark_settings()</code> or a compatible data frame.
future_mode	Future-tile handling passed to <code>autoplay_move()</code> .
seed	Optional positive integer seed for deterministic game seeds. If NULL, a seed is drawn from R's current random-number stream.
workers	Number of PSOCK workers. Use 1 for sequential execution.

Value

A list with summary, runs, settings, config, and seed.

continue_game	<i>Continue after game over</i>
---------------	---------------------------------

Description

Clears the configured number of top rows after a game-over state while preserving score, moves, queue, random-number state, and the largest observed tile used for future spawning.

Usage

```
continue_game(state)
```

Arguments

state	A game-state list returned by new_game() , drop_tile() , or this function.
-------	--

Value

The continued game-state list.

drop_tile	<i>Drop the next tile into a column</i>
-----------	---

Description

Drop the next tile into a column

Usage

```
drop_tile(state, column)
```

Arguments

state	A game-state list returned by new_game() or this function.
column	One-based column index.

Value

The updated game-state list.

export_static_app	<i>Export the standalone static app</i>
-------------------	---

Description

Copies the packaged standalone browser app to an HTML file. The exported app runs entirely in the browser, including game rules, rendering, preview horizon handling, autoplay-lite controls, and browser-local high scores.

Usage

```
export_static_app(path, overwrite = FALSE)
```

Arguments

path	Output HTML file path. This must be supplied explicitly. If path is an existing directory, the file is written as mergeGridR-static.html inside that directory.
overwrite	Whether to replace an existing file.

Value

Invisibly returns the normalized output path.

Examples

```
out <- tempfile(fileext = ".html")
export_static_app(out)
```

game_config	<i>Create Merge Grid game configuration</i>
-------------	---

Description

Create Merge Grid game configuration

Usage

```
game_config(
  rows = 7L,
  cols = 5L,
  next_count = 1L,
  spawn_distribution = c("weighted", "uniform", "weighted_high"),
  spawn_weight_decay = 0.55,
  max_spawn_value = 256L,
  max_continues = 3L,
  continue_clear_rows = 3L
)
```

Arguments

- `rows, cols` Board size. Row 1 is the bottom row.
- `next_count` Preview horizon: number of upcoming tiles kept in the queue.
- `spawn_distribution` Sampling strategy for newly provided tiles.
- `spawn_weight_decay` Geometric weight decay for "weighted" and "weighted_high" spawning. Smaller values more strongly favor the chosen end of the allowed tile range.
- `max_spawn_value` Maximum tile value that can enter the queue.
- `max_continues` Number of manual continues available after game over.
- `continue_clear_rows` Number of top rows cleared by each continue.

Value

A list used by `new_game()`.

<code>get_high_score</code>	<i>Read the local high score</i>
-----------------------------	----------------------------------

Description

Read the local high score

Usage

```
get_high_score(preview_horizon = 1L)
```

Arguments

- `preview_horizon` Preview horizon whose local high score should be read.

Value

A non-negative numeric score. Missing or invalid files return zero.

new_game	<i>Start a new Merge Grid game</i>
----------	------------------------------------

Description

Start a new Merge Grid game

Usage

```
new_game(config = game_config(), seed = NULL)
```

Arguments

config	A configuration list created by <code>game_config()</code> .
seed	Optional positive integer seed for reproducible tile queues.

Value

A game-state list containing the board, score, queue, and status.

reset_high_score	<i>Reset the local high score</i>
------------------	-----------------------------------

Description

Reset the local high score

Usage

```
reset_high_score(preview_horizon = 1L)
```

Arguments

preview_horizon	Preview horizon whose local high score should be reset.
-----------------	---

Value

Invisibly returns zero.

run_drop_number	<i>Launch the mergeGridR Shiny app</i>
-----------------	--

Description

Launch the mergeGridR Shiny app

Usage

```
run_drop_number(  
  host = "127.0.0.1",  
  port = NULL,  
  launch.browser = interactive()  
)
```

Arguments

host Host interface passed to `shiny::runApp()`.
port Optional port passed to `shiny::runApp()`.
launch.browser Passed to `shiny::runApp()`.

Value

The result of `shiny::runApp()`.

Index

* datasets

- autoplay_benchmark_results, [3](#)

- autoplay_benchmark_results, [3](#)
- autoplay_benchmark_settings, [3](#)
- autoplay_benchmark_settings(), [6](#)
- autoplay_game, [4](#)
- autoplay_move, [5](#)
- autoplay_move(), [4](#), [6](#)

- benchmark_autoplay_strategies, [6](#)
- benchmark_autoplay_strategies(), [3](#)

- continue_game, [7](#)

- drop_tile, [7](#)
- drop_tile(), [5](#), [7](#)

- export_static_app, [8](#)

- game_config, [8](#)
- game_config(), [6](#), [10](#)
- get_high_score, [9](#)

- mergeGridR (mergeGridR-package), [2](#)
- mergeGridR-package, [2](#)

- new_game, [10](#)
- new_game(), [5](#), [7](#), [9](#)

- reset_high_score, [10](#)
- run_drop_number, [11](#)

- shiny::runApp(), [11](#)