

Package: chess2plyrs (via r-universe)

May 22, 2026

Type Package

Title Chess Game Creation and Tools

Version 0.3.0

Description A chess program which allows the user to create a game, add moves, check for legal moves and game result, plot the board, take back, read and write FEN (Forsyth–Edwards Notation). A basic chess engine based on minimax is implemented.

Depends R (>= 4.3.0)

Imports stats, ggplot2 (>= 3.4.4)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Repository <https://luigi-annic.r-universe.dev>

Date/Publication 2025-06-06 14:05:00 UTC

RemoteUrl <https://github.com/luigi-annic/chess2plyrs>

RemoteRef HEAD

RemoteSha 3a608da5b93b58c021a3354172b27d3f3e6d8045

Contents

chess_move	2
chessplot	3
chesstools	3
engine2	4
game_result	4

legalmoves	5
moves_scoresheet	5
newgame	6
random_mover	6
readfen	7
takeback	7
writefen	8

Index	9
--------------	----------

chess_move	<i>chess_move</i>
------------	-------------------

Description

Takes in input a move, evaluates whether it is legal, and if it is, then the game is updated

Usage

```
chess_move(game, piece, initialposition = "", finalposition = "")
```

Arguments

game	chess game object (i.e., a list with elements board, turn, history, and fen_history as created by newgame function)
piece	letter indicating the piece to be moved (p, N, B, R, Q, K)
initialposition	initial square of the piece
finalposition	destination square

Value

makes move

Examples

```
newgame() |>
chess_move("N", "g1", "f3")
```

chessplot	<i>chessplot</i>
-----------	------------------

Description

plots the current position

Usage

```
chessplot(game, style = 1)
```

Arguments

game	chess game object (i.e., a list with elements board, turn, history, and fen_history as created by newgame function)
style	font style. 1: chess pieces according to unicode; 2: chess labels

Value

plot

chesstools	<i>chesstools</i>
------------	-------------------

Description

All tools used for letting the chess program work.

Usage

```
chesstools
```

Format

An object of class list

`engine2`*engine2*

Description

engine which chooses minimax between legal moves, with alpha beta pruning

Usage

```
engine2(game, depth)
```

Arguments

<code>game</code>	chess game object (i.e., a list with elements board, turn, history, and fen_history as created by newgame function)
<code>depth</code>	depth of the minimax. depth of 1 and 2 are fairly rapid.

Value

game with new move done

`game_result`*game_result*

Description

This function tells if the game is still ongoing, or if a checkmate or stalemate are on the board

Usage

```
game_result(game)
```

Arguments

<code>game</code>	chess game object (i.e., a list with elements board, turn, history, and fen_history as created by newgame function)
-------------------	---

Value

a message

legalmoves	<i>legalmoves</i>
------------	-------------------

Description

lists legal moves

Usage

```
legalmoves(game)
```

Arguments

game	chess game object (i.e., a list with elements board, turn, history, and fen_history as created by newgame function)
------	---

Value

character vector

Examples

```
newgame() |>  
legalmoves()
```

moves_scoresheet	<i>moves_scoresheet</i>
------------------	-------------------------

Description

Creates move scorelist, in scientific notation

Usage

```
moves_scoresheet(game, shortnotation = TRUE)
```

Arguments

game	chess game object (i.e., a list with elements board, turn, history, and fen_history as created by newgame function)
shortnotation	Use short scientific notation? TRUE is the default

Value

moves scoresheet

Examples

```

g <- newgame() |>
chess_move("p", "e2", "e4") |>
chess_move("p", "e7", "e5") |>
chess_move("N", "g1", "f3") |>
chess_move("N", "b8", "c6") |>
chess_move("B", "f1", "b5") |>
chess_move("N", "g8", "f6") |>
chess_move("K", "e1", "0-0") |>
chess_move("N", "f6", "e4")
moves_scoresheet(g)

```

newgame	<i>newgame</i>
---------	----------------

Description

sets up a new chess game

Usage

```
newgame()
```

Value

new game

random_mover	<i>random_mover</i>
--------------	---------------------

Description

engine which chooses randomly between legal moves

Usage

```
random_mover(game)
```

Arguments

game	chess game object (i.e., a list with elements board, turn, history, and fen_history as created by newgame function)
------	---

Value

game with new move done

readfen	<i>readfen</i>
---------	----------------

Description

read fen (Forsyth–Edwards Notation) notation

Usage

readfen(fenstring)

Arguments

fenstring fen string (pieces and turn)

Value

board and turn

Examples

```
readfen("r3kb1r/pp1nqppp/4bp2/2p5/3P4/1B3N2/PPP1QPPP/R1B1K2R w")
```

takeback	<i>takeback</i>
----------	-----------------

Description

takeback

Usage

takeback(game)

Arguments

game chess game object (i.e., a list with elements board, turn, history, and fen_history as created by newgame function)

Value

game (last move being deleted)

writefen	<i>writefen</i>
----------	-----------------

Description

write fen (Forsyth–Edwards Notation) notation

Usage

```
writefen(game, cb = NULL, tb = NULL, cb_tb_insteadof_game = FALSE)
```

Arguments

game	chess game object (i.e., a list with elements board, turn, history, and fen_history as created by newgame function)
cb	chess board if cb_tf_insteadof_game set to TRUE
tb	turn if cb_tf_insteadof_game set to TRUE
cb_tb_insteadof_game	if FALSE, uses game to create fen, if TRUE it uses cb and tb

Value

fen

Index

* list

- chesstools, 3
- chess_move, 2
- chessplot, 3
- chesstools, 3
- engine2, 4
- game_result, 4
- legalmoves, 5
- moves_scoresheet, 5
- newgame, 6
- random_mover, 6
- readfen, 7
- takeback, 7
- writfen, 8