Package ‘bigchess’

April 3, 2020

Type Package
Title Read, Manipulate, Explore Chess PGN Files and R API to UCI Chess Engines
Version 1.7.1
Author Wojciech Rosa
Maintainer Wojciech Rosa <w.rosa@pollub.pl>
Description Provides functions for reading *.PGN files with more than one game, including large files without copying it into RAM (using 'ff' package or 'RSQLite' package). Handle chess data and chess aggregated data, count figure moves statistics, create player profile, plot winning chances, browse openings. Set of functions of R API to communicate with UCI-protocol based chess engines.
License GPL-3
Imports ff,ffbase,RSQLite,processx
Suggests rjson,magrittr
Encoding UTF-8
LazyData true
RoxygenNote 7.0.0
NeedsCompilation no
Repository CRAN
Date/Publication 2020-04-03 11:30:05 UTC

R topics documented:

analyze_position ................................................................. 2
browse_eco_opening .............................................................. 3
browse_opening ................................................................. 4
eco ................................................................. 5
extract_moves ................................................................. 5
FirstTwoMoves ................................................................. 6
lan2san ................................................................. 7
n_moves ................................................................. 7
analyze_position

Description

Analyze position using UCI engine and R API

Usage

```r
analyze_position(engine, san = NULL, lan = NULL, ...)
```

Arguments

- **engine**: engine object
- **san**: movetext in short algebraic notation, default NULL
- **lan**: movetext in long algebraic notation, default NULL
- **...**: further arguments passed directly to uci_go()
Value

list containg bestomove, score and bestlines

Examples

# Linux (make sure you have executable permission):
engine_path <- "./stockfish_10_x64"
# Windows
# engine_path <- "./stockfish_10_x64.exe"
require(magrittr)
ap <- analyze_position(engine_path,san = "1. e4",depth = 20)
ap$bestmove_lan
# "e7e5"
ap$score
# -7
ap$bestmove_san
# "e5"
ap$curpos_lan
# "e2e4"
ap$curpos_san
# "1. e4"
ap$bestline_lan
# "e7e5 g1f3 b8c6 d2d4 e5d4 f1c4 g8f6 e1g1 f8e7
# f1e1 d7d6 f3d4 c6e5 c4b3 e8g8 b1c3 c7c5 d4f5
# c8f5 e4f5 c5c4 b3a4 a7a6 d1e2"

browse_eco_opening

Browse ECO opening

Description

Browse ECO opening winning and drawing percentages by table and barplot

Usage

browse_eco_opening(df, topn = 0)

Arguments

df data frame with imported chess games from read.pgn() function.
topn integer, default is 0, passed to tree_eco function (indicating how many top openings should be included).
browse_opening

Value

Data frame from tree_eco function and plot from plot_tree_eco function.

Examples

f <- system.file("extdata", "Kasparov.gz", package = "bigchess")
con <- gzfile(f, encoding = "latin1")
df <- read.pgn(con, quiet = TRUE, ignore.other.games = TRUE, stat.moves = FALSE, add.tags = "ECO")
# Analyze 20 best ECO Kasparov openings:
bo <- browse_eco_opening(subset(df, grepl("Kasparov", White)), 20)

browse_opening

Browse opening

Description

Browse opening winning and drawing percentages by table and barplot

Usage

browse_opening(df, movetext = "")

Arguments

df
  data frame with imported chess games from read.pgn() function.
movetext
  movetext string, default is "" means browse first move for White. The standard English values are required: pawn = "P" (often not used), knight = "N", bishop = "B", rook = "R", queen = "Q", and king = "K".

Value

Data frame from tree_move function and plot from plot_tree_move function.

Examples

f <- system.file("extdata", "Kasparov.gz", package = "bigchess")
con <- gzfile(f, encoding = "latin1")
df <- read.pgn(con, quiet = TRUE, ignore.other.games = TRUE, stat.moves = FALSE)
# Analyze best Kasparov openings:
bo <- browse_opening(subset(df, grepl("Kasparov", White)))
# Analyze 'best' answer to Kasparov Ruy Lopez:
bo <- browse_opening(subset(df, grepl("Kasparov", White)), "1.e4 e5 2.Nf3 Nc6 3.Bb5")
# Analyze best answer to "1.e4 e5 2.Nf3" in aggregated data
browse_opening(FirstTwoMoves, "1.e4 e5 2.Nf3")
**Description**

A dataset containing 2014 ECO (Encyclopedia of Chess Openings) openings

- ECO
- Opening
- Variation
- Movetext: Standard algebraic notation
- NMoves
- LAN: Movetext converted into long algebraic notation

**Usage**

data(eco)

**Format**

A data frame with ECO openings

---

**extract_moves**

*Extract first N moves*

**Description**

Extract first N moves from pgn movetext into data frame

**Usage**

extract_moves(movetext, N = 10, last.move = T)

**Arguments**

- **movetext**
  movetext string (or string vector). The standard English values are required: pawn = "P" (often not used), knight = "N", bishop = "B", rook = "R", queen = "Q", and king = "K".

- **N**
  integer (default 10) determines how many first N moves will be extracted. Default is 10, should be greater than 0.

- **last.move**
  boolean (default TRUE) indicating whether to calculate the last move
Value

Data frame containing first \( N \) moves for white and for black, named as \( W_1, B_1, W_2 \) and so on, up to \( W_N \) and \( B_N \) (where \( N \) is input argument). If \( N \) is greater than total moves number then NA’s generated. Column complete.movetext flag is indicating if movetext string begin with "1.'move'".

Examples

```r
extract_moves("1. e4 e5 2. Nf3 Nf5 3. d5 ", N = 3)
# e4 e5 Nf3 Nf5 d5 NA TRUE
extract_moves("1. e4 e5 2. Nf3 Nf5 3. d5 ", N = 3, last.move = TRUE)
# e4 e5 Nf3 Nf5 d5 NA d5 TRUE
```

Description

A dataset containing 10,894 results after first two moves in 2,395,869 high-quality chess games played over the board by players with ELO > 2000. Source data OTB-HQ.7z downloaded from: https://sourceforge.net/projects/codekiddy-chess/ and converted to PGN in SCID software.

- Result:
  - W1: White first move
  - B1: Black first move
  - W2: White second move
  - B2: Black second move
  - Freq: Number of games played in database

Usage

```r
data(FirstTwoMoves)
```

Format

A data frame with popular positions in classic chess
lan2san

Movetext conversion from LAN to SAN

Description
Convert LAN movetext (long algebraic notation used by chess engines) to SAN movetext (standard algebraic notation required by FIDE)

Usage
lan2san(movetext.lan)

Arguments
movetext.lan movetext string in long algebraic notation (LAN), but without comments, variants etc.

Value
movetext in standard algebraic notation

Examples
lan2san("e2e4 c7c5")

n_moves
Compute number of moves

Description
Compute total number of moves given movetext string (or string vector)

Usage
n_moves(movetext)

Arguments
movetext movetext string (or string vector)

Value
n integer (or integer vector)

Examples
n_moves(c("1. e4 e5 2. Nf3 Nf5 3. d5 ","1. d4 d5"))
# 3 1
player_profile  Compute player profile

Description

Computes players profile from data frame obtained from read.pgn() function into data frame

Usage

player_profile(df, player)

Arguments

df  data frame from read.pgn or read.pgn.ff files with stats computed.
player  string used in grepl(player,White) and grepl(player,Black)

Value

Data frame with player (column prefix P_) and opponent (column prefix O_) figure move counts. Column Player_Col indicating pieces colour for player (factor White or Black). Example column P_Q_moves means number of player Queen moves count.

Examples

f <- system.file("extdata", "Kasparov.gz", package = "bigchess")
con <- gzfile(f,encoding = "latin1")
df <- read.pgn(con,quiet = TRUE,ignore.other.games = TRUE)
nrow(df) # 2109
df_pp <- player_profile(df,"Kasparov, Gary")
nrow(df_pp) # 1563
df_pp <- player_profile(df,"Kasparov,G")
nrow(df_pp) # 543
df_pp <- player_profile(df,"Kasparov, G\.")
nrow(df_pp) # 2
df_pp <- player_profile(df,"Kasparov")
nrow(df_pp) # 2109 - correct
boxplot(P_Q_moves/NMoves~Player_Col,df_pp,
main = "Average Queen Moves\n Kasparov as Black (909 games) vs Kasparov as White (1200 games)",
col = c("black","white"),border = c("black","black"),notch = TRUE)
# Magnus Carlsen data example
f <- system.file("extdata", "Carlsen.gz", package = "bigchess")
con <- gzfile(f,encoding = "latin1")
df <- read.pgn(con,quiet = TRUE,ignore.other.games = TRUE)
nrow(df) # 2410
df_pp <- player_profile(df,"Carlsen")
nrow(df_pp) # 2411 - ??
# One game was played by Carlsen,H
df_pp <- player_profile(df,"Carlsen,H")
nrow(df_pp) # 2410 - correct
plot_tree_eco  

*Plot tree for a given tree ECO table*

**Description**

Plot tree (barplot percentages) for a given tree ECO data frame.

**Usage**

```r
plot_tree_eco(tr, main = "", add.lines = T, add.labels = T)
```

**Arguments**

- `tr`: data frame containing tree ECO
- `main`: string for main title, default is ""
- `add.lines`: boolean (default TRUE) to add weighted mean lines?
- `add.labels`: boolean (default TRUE) to add labels?

**Value**

Barplot with white scores, draws percent and black scores.

**Examples**

```r
f <- system.file("extdata", "Kasparov.gz", package = "bigchess")
con <- gzfile(f, encoding = "latin1")
df <- read.pgn(con, quiet = TRUE, stat.moves = FALSE, add.tags = "ECO")
tr <- tree_eco(subset(df, W1 == "e4"), 20)
plot_tree_eco(tr, "1. e4 ... ?")
```

---

plot_tree_move  

*Plot tree for a given tree move table*

**Description**

Plot tree (barplot percentages) for a given tree move data frame.

**Usage**

```r
plot_tree_move(tr, main = "", add.lines = T, add.labels = T)
```

**Arguments**

- `tr`: data frame containing tree move
- `main`: string for main title, default is ""
- `add.lines`: boolean (default TRUE) to add weighted mean lines?
- `add.labels`: boolean (default TRUE) to add labels?
**Value**

Barplot with white scores, draws percent and black scores.

**Examples**

```r
f <- system.file("extdata", "Kasparov.gz", package = "bigchess")
con <- gzfile(f, encoding = "latin1")
df <- read.pgn(con, quiet = TRUE, stat.moves = FALSE)
tr <- tree_move(subset(df, W1 == "e4"), "B1")
plot.tree.move(tr, "1. e4 ... ?")
# Plot tree move openings in aggregated data
tr <- tree_move(FirstTwoMoves, "W1")
plot.tree.move(tr, paste0("1. ... ?\n", sum(FirstTwoMoves$Freq), " total games"))
```

---

**read.pgn**

*Reads PGN files into data frame*

**Description**

Reads PGN files into data frame

**Usage**

```r
read.pgn(
  con,
  add.tags = NULL,
  n.moves = T,
  extract.moves = 10,
  last.move = T,
  stat.moves = T,
  big.mode = F,
  quiet = F,
  ignore.other.games = F,
  source.movetext = F
)
```

**Arguments**

- **con**: connection argument passed directly to readLines() function. String - the name of the file which the data are to be read from or connection object or URL.
- **add.tags**: string vector containing additional tags to be parsed. According to Seven Tag Roster rule: http://www.saremba.de/chessgml/standards/pgn/pgn-complete.htm#c8.1.1 The STR tag pairs appear before any other tag pairs: "Event", "Site", "Date", "Round", "White", "Black" and "Result". Using this argument you can specify supplemental tag names, such as: Player related information, Event related information, Opening information (locale specific), Opening information...
(third party vendors), Time and date related information, Time control, Alternative starting positions, Game conclusion and Miscellaneous. Most popular: "WhiteElo", "BlackElo", "ECO", "SetUp" or "FEN". Case sensitive.

- **n.moves**: boolean (default TRUE), compute number of moves?
- **extract.moves**: integer (default 10) passed to extract_moves function. Additionally value -1 will extract all moves from movetext (not recommended for big files). Value 0 means that moves will not be extracted.
- **last.move**: boolean (default TRUE) passed to extract_moves, ignored when extract.moves = 0
- **stat.moves**: boolean (default TRUE), compute moves count statistics? Could take a long time for big file.
- **big.mode**: boolean (default FALSE) used in read.pgn.ff function
- **quiet**: boolean (default FALSE), indicating if messages should appear.
- **ignore.other.games**: boolean (default FALSE) if TRUE result is subset of original dataset without games with result marked as "+", i.e. ongoing games
- **source.movetext**: boolean (default FALSE, experimental!) if TRUE column with original movetext will be added

**Value**

Data frame containing STR, additional tags (conditionally), Movetext, NMoves (conditionally), extracted moves (conditionally) with complete.movetext flag, figure moves count statistics (conditionally).

**Examples**

```r
def <- system.file("extdata", "2016_Candidates.pgn", package = "bigchess")
df <- read.pgn(f)
# ...successfully imported 56 games...

# Example downloaded from https://www.pgnmentor.com/files.html#players and gzipped:
f <- system.file("extdata", "Carlsen.gz", package = "bigchess")
con <- gzfile(f, encoding = "latin1")
df <- read.pgn(con, quiet = TRUE)
# Fastest mode:
con <- gzfile(f, encoding = "latin1")
df <- read.pgn(con, quiet = TRUE, n.moves = FALSE, extract.moves = FALSE, stat.moves = FALSE, ignore.other.games = FALSE)
# Parse additional tags and extract all moves:
con <- gzfile(f, encoding = "latin1")
df <- read.pgn(con, add.tags = c("WhiteElo", "BlackElo", "ECO"), extract.moves = -1)
# Example of direct downloading data from chess.com using API:
df <- read.pgn("https://api.chess.com/pub/player/fabianocaruana/games/2013/03/pgn")
# Warning of incomplete line could appear

# Example of scraping all of games given user:
```

user <- "fabianocaruana"
library("rjson")
json_file <- paste0("https://api.chess.com/pub/player/", user, "/games/archives")
json_data <- fromJSON(paste(readLines(json_file), collapse=""))
result <- data.frame()
for(i in json_data$archives)
  result <- rbind(result, read.pgn(paste0(i,"/pgn")))

---

**read.pgn.db**

**Reads PGN files into database table**

**Description**

Reads PGN files into database table

**Usage**

read.pgn.db(con, batch.size = 10^6, conn, table.name = "pgn", ...)

**Arguments**

- **con**: connection argument passed directly to readLines() function. String - the name of the file which the data are to be read from or connection object or URL.
- **batch.size**: number of lines to read in one batch, default is 10^6.
- **conn**: connection argument created by dbConnect
- **table.name**: string (default "pgn"), table name, used in dbWriteTable(conn, table.name, read.pgn(batch))
- **...**: further arguments passed directly to read.pgn() function (besides ignore.other.games and big.mode)

**Examples**

```r
f <- system.file("extdata", "Carlsen.gz", package = "bigchess")
con <- gzfile(f,"rb",encoding = "latin1")
require(RSQLite)
con <- dbConnect(SQLite())
read.pgn.db(con,stat.moves = FALSE,conn = conn)
dbGetQuery(conn, "SELECT COUNT(*) FROM pgn") #2410
dbDisconnect(conn)
# Works with all types of connections (also gz or zip files).
# con argument is passed directly to readLines(con,batch.size)
# so (if total number of lines to read is greater then batch.size)
# depending on platform use it correctly:

# Windows ('rb' opening mode for loop over readLines):
con <- gzfile(system.file("extdata", "Carlsen.gz", package = "bigchess"),"rb",encoding = "latin1")
# con <- file("path_to_big_chess_file.pgn","rb",encoding = "latin1")
read.pgn.db(con,conn = conn)
```
# Linux/Mac OS X ('r' opening mode for loop over readLines):  
con <- gzfile(system.file("extdata", "Carlsen.gz", package = "bigchess"), "r", encoding = "latin1")  
# con <- file("path_to_big_chess_file.pgn", "r", encoding = "latin1")  
read.pgn.db(con, conn = conn)

# Windows (example of zipped file handling)  
unzf <- unzip("zipped_pgn_file.zip")  
read.pgn.db(con, conn = conn)

---

`read.pgn.ff`    *Reads PGN files into ff data frame*

**Description**

Reads PGN files into ff data frame.

**Usage**

`read.pgn.ff(con, batch.size = 10^6, ignore.other.games = F, ...)`

**Arguments**

- `con`: connection argument passed directly to `readLines()` function. String - the name of the file which the data are to be read from or connection object or URL.
- `batch.size`: number of lines to read in one batch, default is 10^6.
- `ignore.other.games`: boolean (default FALSE) if TRUE result is subset of original dataset without games with result marked as "*", i.e. ongoing games. The only one argument which is not passed directly to `read.pgn` function.

**Value**

ff data frame like from `read.pgn()` function. Since character values are not supported in `ffdf` object, "Movetext" column is omitted.

**Examples**

```r
require(ff)  
require(ffbase)  
f <- system.file("extdata", "Carlsen.gz", package = "bigchess")  
con <- gzfile(f, "rbt", encoding = "latin1")  
# options("ftempdir"="/path/"...) # if necessarily  
fd <- read.pgn.ff(con, stat.moves = FALSE)
```
san2lan

 Movetext conversion from SAN to LAN

Description

Convert SAN movetext (FIDE) to LAN movetext (used by chess engines)

Usage

san2lan(movetext.san)

Arguments

movetext.san movetext string in standard algebraic notation (SAN) required by FIDE, but without comments, variants etc.

Value

movetext in long algebraic notation

Examples

san2lan("1. e4 e5 2. Nf3 Nf5 3. d5 ")
**stat_moves**

*Extract statistics of moves*

**Description**

Extract statistics of moves (counts figure moves) from movetext vector into data frame

**Usage**

```r
stat_moves(movetext, sides = "both")
```

**Arguments**

- `movetext`: movetext string (or string vector). The standard English values are required: pawn = "P" (often not used), knight = "N", bishop = "B", rook = "R", queen = "Q", and king = "K".
- `sides`: "both" (default), "white" or "black"

**Value**

Data frame containing moves count statistics for white and for black and total.

**Examples**

```r
stat_moves("1. e4 e5 2. Nf3 Nf5 3. d5 ")
```

---

**tree_eco**

*Compute ECO tree*

**Description**

Compute ECO tree (frequencies and winning percent)

**Usage**

```r
tree_eco(df, topn = 0)
```

**Arguments**

- `df`: data frame containing ECO and Result columns
- `topn`: integer, default 0, indicating how many top openings should be included, 0 means show all openings
Value

Data frame containing White_score (White winning percent), Draws_percent, Black_score and N (number of games). Sorted by power of ECO (White_score * N which describes popularity and score of move) descending.

Examples

```r
f <- system.file("extdata", "Kasparov.gz", package = "bigchess")
con <- gzfile(f, encoding = "latin1")
df <- read.pgn(con, quiet = TRUE, stat.moves = FALSE, add.tags = "ECO")
```

---

## tree_move

*Compute tree for a given move*

### Description

Compute tree for a given move (frequencies and winning percent)

### Usage

```r
tree_move(df, move)
```

### Arguments

- **df**: data frame containing move and Result column from pgn function or data frame containing aggregated data from such df (containing columns: Result, \(W_1\), \(B_1\), \(W_2\), ..., \(W_N\), \(B_N\), Freq)
- **move**: character indicating which move should be browsed, example "W1"

### Value

Data frame containing White_score (White winning percent), Draws_percent, Black_score and N (number of games). Sorted by power of move (White_score times N which describes popularity and score of move) descending.

### Examples

```r
f <- system.file("extdata", "Kasparov.gz", package = "bigchess")
con <- gzfile(f, encoding = "latin1")
df <- read.pgn(con, quiet = TRUE, stat.moves = FALSE)
# Analyze best answers to 1. e4 in Kasparov games (both white and black)
tree_move(subset(df, W1 == "e4"), move = "B1")
# Analyze openings in aggregated data
tree_move(FirstTwoMoves, "W1")
```
uci_cmd

Sending command to chess engine

Description

Sending command to chess engine

Usage

uci_cmd(engine, command = "")

Arguments

engine  
engine object

command  
string command

Value

engine object

Examples

# Linux (make sure you have executable permission):
engine_path <- "./stockfish_10_x64"

# Windows
# engine_path <- "./stockfish_10_x64.exe"
e <- uci_engine(engine_path)
e <- uci_command(e,"go depth 10")
uci_quit(e)

# Using pipe '%>%' from magrittr:
require(magrittr)
uci_engine(engine_path) %>% uci_command("go depth 10") %>% uci_quit()

uci_debug

Sending command debug for chess engine

Description


switch the debug mode of the engine on and off. In debug mode the engine should sent additional infos to the GUI, e.g. with the "info string" command, to help debugging, e.g. the commands that the engine has received etc. This mode should be switched off by default and this command can be sent any time, also when the engine is thinking.
Usage

uci_debug(engine, on = TRUE)

Arguments

engine  
engine object

on  
boolean default TRUE

Value

engine object

uci_engine  Create an engine handler in R

Description

Create an engine handler in R and send command isready

Usage

uci_engine(path)

Arguments

path  
path to engine file. Make sure you have executable permission on this file.

Value

engine object (list of two: pipe to engine and temp as a result from stdout engine)

Examples

# Linux (make sure you have executable permission):
engine_path <- "./stockfish_10_x64"
# Windows
# engine_path <- "./stockfish_10_x64.exe"
e <- uci_engine(engine_path)
uci_quit(e)

# Using pipe '>%>' from magrittr:
require(magrittr)
uci_engine(engine_path) %> uci_quit()
start calculating on the current position set up with the "position" command. There are a number of
commands that can follow this command, all will be sent in the same string. If one command is not
send its value should be interpreted as it would not influence the search.

Usage

uci_go(
  engine,
  depth = NULL,
  infinite = FALSE,
  stoptime = 1,
  wtime = NULL,
  btime = NULL,
  winc = NULL,
  binc = NULL
)

Arguments

engine  engine object
depth   integer depth (search x plies only)
infinite boolean default FALSE. If TRUE, stoptime (next argument) should be defined
stoptime integer default 1. Used in Sys.sleep after go infinite in engine. After this, uci_stop() is executed
wtime   integer default NULL (white has x msec left on the clock)
btime   integer default NULL (black has x msec left on the clock)
winc    integer default NULL (white increment per move in mseconds if x > 0)
binc    integer default NULL (black increment per move in mseconds if x > 0)

Value

engine object

Examples

# Linux (make sure you have executable permission):
engine_path <- "./stockfish_10_x64"
# Windows
# engine_path <- "./stockfish_10_x64.exe"
uci_isready

Checking if chess engine is ready

Description

Checking if chess engine is ready - sending command isready and parsing GUI until readyok is obtained. Info about isready command from http://wbec-ridderkerk.nl/html/UCIProtocol.html This is used to synchronize the engine with the GUI. When the GUI has sent a command or multiple commands that can take some time to complete, this command can be used to wait for the engine to be ready again or to ping the engine to find out if it is still alive. E.g. this should be sent after setting the path to the tablebases as this can take some time. This command is also required once before the engine is asked to do any search to wait for the engine to finish initializing. This command must always be answered with "readyok" and can be sent also when the engine is calculating in which case the engine should also immediately answer with "readyok" without stopping the search.

Usage

uci_isready(engine)

Arguments

engine

engine object

Value

engine object

Examples

# Linux (make sure you have executable permission):
engine_path <- "./stockfish_10_x64"
# Windows
# engine_path <- "./stockfish_10_x64.exe"
e <- uci_engine(engine_path)
e <- uci_isready(e)
uci_parse

uci_quit(e)
# Using pipe '%>%' from magrittr:
require(magrittr)
uci_engine(engine_path) %>% uci_isready() %>% uci_quit()

uci_parse Parse GUI commands from chess engine

Description

Parse GUI commands from chess engine.

Usage

uci_parse(ucilog, filter = "bestmove")

Arguments

ucilog strings from uci_quit() or uci_read()$temp
filter string, one of 'bestmove' (default), 'score' or 'bestline'

Value

strings with parsed information from engine

Examples

# Linux (make sure you have executable permission):
engine_path <- "/stockfish_10_x64"
# Windows
# engine_path <- "/stockfish_10_x64.exe"
require(processx)
e <- uci_engine(engine_path)
e <- uci_go(depth = 10)
rslt <- uci_quit(e)
uci_parse(rslt)
# Using pipe '%>%' from magrittr:
require(magrittr)
uci_engine(engine_path) %>% uci_go(depth = 10) %>% uci_quit() %>% uci_parse()
**uci_ponderhit**  
*Sending command ponderhit for chess engine*

**Description**

Sending command ponderhit for chess engine. Info about ponderhit command from http://wbec-ridderkerk.nl/html/UCIProtocol.html the user has played the expected move. This will be sent if the engine was told to ponder on the same move the user has played. The engine should continue searching but switch from pondering to normal search.

**Usage**

```r
uci_ponderhit(engine)
```

**Arguments**

- **engine**  
  engine object

**Value**

- engine object

---

**uci_position**  
*Sending command position for chess engine*

**Description**

Sending command position for chess engine. Info about position command from http://wbec-ridderkerk.nl/html/UCIProtocol.html set up the position described in fenstring on the internal board and play the moves on the internal chess board. if the game was played from the start position the string "startpos" will be sent. Note: no "new" command is needed. However, if this position is from a different game than the last position sent to the engine, the GUI should have sent a "ucinewgame" in between.

**Usage**

```r
uci_position(engine, moves = NULL, startpos = TRUE, fen = NULL)
```

**Arguments**

- **engine**  
  engine object
- **moves**  
  string in long algebraic notation
- **startpos**  
  boolean default TRUE
- **fen**  
  string
Value

engine object

Examples

# Linux (make sure you have executable permission):
engine_path <- "/stockfish_10_x64"
# Windows
# engine_path <- "/stockfish_10_x64.exe"
e <- uci_engine(engine_path)
e <- uci_position(e, moves = "e2e4")
e <- uci_go(e, depth = 10)
uci_quit(e)
# Using pipe '%>' from magrittr:
require(magrittr)
uci_engine(engine_path) %>% uci_position(moves = "e2e4") %>%
uci_go(depth = 10) %>% uci_quit() %>% uci_parse()

uci_quit

Sending quit command to chess engine

Description

Sending quit command to chess engine and cleaning temps from R

Usage

uci_quit(engine)

Arguments

description

engine engine object

Value

strings from uci chess engine GUI

Examples

# Linux (make sure you have executable permission):
engine_path <- "./stockfish_10_x64"
# Windows
# engine_path <- "/stockfish_10_x64.exe"
e <- uci_engine(engine_path)
uci_quit(e)
# Using pipe '%>' from magrittr:
require(magrittr)
uci_engine(engine_path) %>% uci_quit()
uci_register

Sending command register for chess engine. Info about register command from http://wbec-ridderkerk.nl/html/UCIProtocol.html this is the command to try to register an engine or to tell the engine that registration will be done later. This command should always be sent if the engine has send "registration error" at program startup.

Usage

uci_register(engine, later = TRUE, name = NULL, code = NULL)

uci_read

Read current stdout from chess engine

Description

Read current stdout from chess engine

Usage

uci_read(engine)

Arguments

engine engine object

Value

engine object

Examples

# Linux (make sure you have executable permission):
engine_path <- "./stockfish_10_x64"
# Windows
# engine_path <- "./stockfish_10_x64.exe"
e <- uci_engine(engine_path)
e <- uci_read(e)
e$temp
uci_quit(e)
Arguments

- engine: engine object
- later: boolean default TRUE
- name: string
- code: string

Value

- engine object

Description

Sending command setoption for chess engine. Info about setoption command from http://wbec-ridderkerk.nl/html/UCIProtocol.html this is sent to the engine when the user wants to change the internal parameters of the engine. For the "button" type no value is needed. One string will be sent for each parameter and this will only be sent when the engine is waiting. The name of the option in should not be case sensitive and can includes spaces like also the value. The substrings "value" and "name" should be avoided in and to allow unambiguous parsing, for example do not use = "draw value".

Usage

uci_setoption(engine, name = NULL, value = NULL)

Arguments

- engine: engine object
- name: string option name
- value: string option value

Value

- engine object
uci_stop  Sending command stop for chess engine

Description

Sending command stop for chess engine. Info about stop command from http://wbec-ridderkerk.nl/html/UCIProtocol.html stop calculating as soon as possible, don’t forget the "bestmove" and possibly the "ponder" token when finishing the search

Usage

uci_stop(engine)

Arguments

engine  engine object

Value

engine object

Examples

# Linux (make sure you have executable permission):
engine_path <- "./stockfish_10_x64"
# Windows
# engine_path <- "./stockfish_10_x64.exe"
e <- uci_engine(engine_path)
e <- uci_go(depth = 100)
Sys.sleep(1)
e <- uci_stop(e)
ucci_quit(e)

uci_ucci  Sending command uci for chess engine

Description

Sending command uci for chess engine. Info about uci command from http://wbec-ridderkerk.nl/html/UCIProtocol.html tell engine to use the uci (universal chess interface), this will be send once as a first command after program boot to tell the engine to switch to uci mode. After receiving the uci command the engine must identify itself with the "id" command and sent the "option" commands to tell the GUI which engine settings the engine supports if any. After that the engine should sent "uciok" to acknowledge the uci mode. If no uciok is sent within a certain time period, the engine task will be killed by the GUI.
Usage
uci_ucc(eng)ne

Arguments

engineengine object

Value

eengine object

Sending command ucinewgame for chess engine

Description

Sending command ucinewgame for chess engine. Info about ucinewgame command from http://wbecridderkerk.nl/html/UCIProtocol.html this is sent to the engine when the next search (started with "position" and "go") will be from a different game. This can be a new game the engine should play or a new game it should analyse but also the next position from a testsuite with positions only. If the GUI hasn't sent a "ucinewgame" before the first "position" command, the engine shouldn't expect any further ucinewgame commands as the GUI is probably not supporting the ucinewgame command. So the engine should not rely on this command even though all new GUIs should support it. As the engine's reaction to "ucinewgame" can take some time the GUI should always send "isready" after "ucinewgame" to wait for the engine to finish its operation.

Usage
uci_ucc(eng)ne

Arguments

engineengine object

Value

eengine object
Index

*Topic datasets
  eco, 5
  FirstTwoMoves, 6
analyze_position, 2
browse_eco_opening, 3
browse_opening, 4
eco, 5
extract_moves, 5
FirstTwoMoves, 6
lan2san, 7
n_moves, 7
player_profile, 8
plot_tree_eco, 9
plot_tree_move, 9
read.pgn, 10
read.pgn.db, 12
read.pgn.ff, 13
san2lan, 14
stat_moves, 15
tree_eco, 15
tree_move, 16
uci_cmd, 17
uci_debug, 17
uci_engine, 18
uci_go, 19
uci_isready, 20
uci_parse, 21
uci_ponderhit, 22
uci_position, 22
uci_quit, 23
uci_read, 24
uci_register, 24
uci_setoption, 25
uci_stop, 26
uci_uuci, 26
uci_uicinewgame, 27