

Package: ppdf (via r-universe)

October 10, 2026

Encoding UTF-8

Type Package

Title Board Game Setups

Version 0.3.0-20

Description Generate 'tibble' data frames indicating how to set up over a hundred board games playable with public domain game systems. Data format can be used by 'piecepackr' to generate graphics with the 'grid', 'ggplot2', 'rayrender', 'rayvertex', and 'rgl' graphics systems and by 'ppcli' to generate plaintext graphics with 'cli'.

License MIT + file LICENSE

URL <https://github.com/piecepackr/ppdf>,
<https://trevorldavis.com/R/ppdf/>

BugReports <https://github.com/piecepackr/ppdf/issues>

LazyLoad yes

Depends R (>= 4.1.0)

Imports dplyr, purrr, rlang, stringr, tibble, tidyr, utils, withr,

Suggests knitr, piecepackr (>= 1.16.0-3), ppcli (>= 0.3.0-2), snakecase, systemfonts, testthat, vdiffr

Remotes piecepackr/piecepackr, piecepackr/ppcli

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-05-01 06:50:54 UTC

RemoteUrl <https://github.com/piecepackr/ppdf>

RemoteRef HEAD

RemoteSha 5261c58e168cfc2d1d3208c09437d824ed32a531

Contents

alquerque_games_variant	3
alquerque_pieces	4
checker_games_other	5
checker_games_variant	7
checker_pieces	9
chess_games_variant	11
chess_pieces	13
cube_pieces	15
dice_pieces	17
domino_games_variant	20
domino_tiles	22
games_info	23
go_games_variant	24
go_pieces	25
marble_pieces	27
meeple_pieces	29
morris_games_variant	30
morris_pieces	31
normalize_name	33
piece_angle	33
piece_rank	34
piece_suit	35
piecepack_board	36
piecepack_games_checkers	37
piecepack_games_chess	39
piecepack_games_original	41
piecepack_games_other	46
piecepack_games_traditional	48
piecepack_pieces	50
piecepack_preview	53
ppdf-defunct	54
reversi_games_variant	58
reversi_pieces	59
setup_by_name	60
setup_none	63
stackpack_games_other	64
tarot_pieces	65

Index

68

alquerque_games_variant

Setups for alquerque variants

Description

tibble data frames of setups for 2 alquerque variants. Data frame output can usually be plotted with `ppcli::cat_piece(df, default.units = "in")`.

Usage

```
alquerque_alquerque()
```

```
alquerque_baghchal()
```

Details

Here are links for more information about the various games:

Game	URL
Alquerque	https://en.wikipedia.org/wiki/Alquerque
Bagh-chal	https://en.wikipedia.org/wiki/Bagh-chal

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- alquerque_alquerque()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.2.0-1") {
  ppcli::cat_piece(df)
}
```

alquerque_pieces	<i>Generate alquerque boards and pieces</i>
------------------	---

Description

alquerque_board() returns an alquerque board. alquerque_bits() returns alquerque pieces.

Usage

```
alquerque_board(x0 = 1, y0 = 1, ..., suit = "green")
```

```
alquerque_bits(
  ...,
  suit = 1:6,
  x = as.double(1:6),
  y = 1,
  angle = 0,
  length.out = NA_integer_
)
```

Arguments

x0	X coordinate for the center of the first point
y0	Y coordinate for the center of the first point
...	Should be left empty.
suit	Suit value (color) of board/bit. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by piece_suit() .
x, y	Cartesian coordinates (numeric vectors)
angle	Rotation of piece (numeric vector of degrees, counter-clockwise). Will be coerced by piece_angle() .
length.out	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A [tibble::tibble\(\)](#) data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with [piecepackr::render_piece\(\)](#) and likely [ppcli::cat_piece\(\)](#).

Examples

```
df <- alquerque_board()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.2.0-1") {
  ppcli::cat_piece(df)
}
```

checker_games_other *Setups for other games playable with a checkers set*

Description

tibble data frames of setups for 11 other games playable with a checkers set. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
checker_breakthrough(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_crossings(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_dao(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_dodgem(
  nrows = 3L,
  cell_width = getOption("ppdf.checker_cell_width", 1)
)

checker_focus(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_four_field_kono(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_grasshopper(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_jeson_mor(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_julgonu(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_kangaroo(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_lines_of_action(cell_width = getOption("ppdf.checker_cell_width", 1))
```

Arguments

<code>cell_width</code>	Width of board cell. Most renderers support 1 or 2.
<code>nrows</code>	Number of rows (and columns) in game board.

Details

Here are links for more information about the various games:

Game	URL
Breakthrough	https://en.wikipedia.org/wiki/Breakthrough_(board_game)
Crossings	https://en.wikipedia.org/wiki/Crossings_(game)
Dao	https://boardgamegeek.com/boardgame/948/dao
Dodgem	https://en.wikipedia.org/wiki/Dodgem
Focus	https://en.wikipedia.org/wiki/Focus_(board_game)
Four Field Kono	https://en.wikipedia.org/wiki/Four_Field_Kono
Grasshopper	http://www.cyclingstan.com/game/71/grasshopper
Jeson Mor	https://en.wikipedia.org/wiki/Jeson_Mor
Jul-Gonu	https://en.wikipedia.org/wiki/Jul-Gonu
Kangaroo	https://boardgamegeek.com/boardgame/6132/kangaroo-the-jumping-game
Lines of Action	https://en.wikipedia.org/wiki/Lines_of_Action

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- checker_four_field_kono()
if (require("piecepackr", quietly = TRUE) &&
    packageVersion("piecepackr") >= "1.15.0-1") {
  grid::grid.newpage()
  render_piece(df, open_device = FALSE, annotate = TRUE)
}
```

checker_games_variant *Setups for checkers variants*

Description

tibble data frames of setups for 23 checkers variants. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
checker_american_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_american_pool_checkers(
  cell_width = getOption("ppdf.checker_cell_width", 1)
)

checker_bashni(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_brazilian_checkers(
  cell_width = getOption("ppdf.checker_cell_width", 1)
)

checker_canadian_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_column_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_corner_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_czech_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_dameo(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_english_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_frisian_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_gothic_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_international_checkers(
  cell_width = getOption("ppdf.checker_cell_width", 1)
)

checker_italian_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_jamaican_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))
```

```

checker_lasca(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_malaysian_checkers(
  cell_width = getOption("ppdf.checker_cell_width", 1)
)

checker_one_way_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_portuguese_checkers(
  cell_width = getOption("ppdf.checker_cell_width", 1)
)

checker_russian_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_singaporean_checkers(
  cell_width = getOption("ppdf.checker_cell_width", 1)
)

checker_spanish_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_thai_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_turkish_checkers(cell_width = getOption("ppdf.checker_cell_width", 1))

checker_unified_pool_checkers(
  cell_width = getOption("ppdf.checker_cell_width", 1)
)

checker_zimbabwean_pool_checkers(
  cell_width = getOption("ppdf.checker_cell_width", 1)
)

```

Arguments

`cell_width` Width of board cell. Most renderers support 1 or 2.

Details

Here are links for more information about the various games:

Game

(American) Checkers AKA (English) Draughts
 American Pool Checkers
 Bashni AKA Column Checkers
 Brazilian Checkers AKA Brazilian Draughts
 Canadian Checkers AKA Canadian Draughts
 Corner Checkers AKA Corner Draughts
 Czech Checkers AKA Czech Draughts

URL

<https://en.wikipedia.org/wiki/Draughts>
<https://draughts.github.io/american-pool-checkers/>
<https://en.wikipedia.org/wiki/Bashni>
https://en.wikipedia.org/wiki/Brazilian_draughts
https://en.wikipedia.org/wiki/Canadian_checker
<https://brainking.com/en/GameRules?tp=30>
https://en.wikipedia.org/wiki/Czech_draughts

Dameo	https://en.wikipedia.org/wiki/Dameo
Frisian Checkers AKA Frisian Draughts	https://en.wikipedia.org/wiki/Frisian_draughts
Gothic Checkers AKA Gothic Draughts	http://mlwi.magix.net/bg/gothiccheckersvariant
International Checkers AKA International Draughts	https://en.wikipedia.org/wiki/International_dr
Italian Checkers AKA Italian Draughts	https://en.wikipedia.org/wiki/Italian_draughts
Jamaican Checkers AKA Jamaican Draughts	https://web.archive.org/web/20230605023244/http://www.lasca.org
Lasca	http://www.lasca.org
Malaysian Checkers AKA Singaporean Checkers	https://brainking.com/en/GameRules?tp=31
One Way Checkers AKA One Way Draughts	https://brainking.com/en/GameRules?tp=31
Portuguese Checkers AKA Portuguese Draughts	http://www.fpdamas.pt/regras/
Russian Checkers AKA Russian Draughts	https://en.wikipedia.org/wiki/Russian_draughts
Spanish Checkers AKA Spanish Draughts	https://mindsports.nl/index.php/on-the-evoluti
Thai Checkers AKA Thai Draughts AKA Mak-hot AKA Makhos	https://checkers.fandom.com/wiki/Mak-hot
Turkish Checkers AKA Turkish Draughts AKA Dama	https://en.wikipedia.org/wiki/Turkish_draughts
Unified Pool Checkers	https://wiegerw.github.io/pdn/gametype.html
Zimbabwean Pool Checkers	https://wiegerw.github.io/pdn/gametype.html

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- checker_american_checkers()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

checker_pieces

Generate checkers boards and pieces

Description

`checker_board()` returns a square checkers board of desired size and type. `checker_bits()` returns checkers discs.

Usage

```
checker_board(
  nrows = 8L,
  ncols = nrows,
  x0 = 1,
  y0 = 1,
  ...,
  side = "face",
  piece_side = paste0("board_", side),
  suit = "green",
  angle = 0,
  cell_width = getOption("ppdf.checker_cell_width", 1)
)
```

```
checker_bits(
  ...,
  side = "back",
  piece_side = paste0("bit_", side),
  suit = 1:6,
  x = as.double(1:6),
  y = 1,
  angle = 0,
  cell_width = getOption("ppdf.checker_cell_width", 1),
  length.out = NA_integer_
)
```

Arguments

nrows	Number of rows in game board
ncols	Number of columns in game board
x0	X coordinate for the center of the first cell
y0	Y coordinate for the center of the first cell
...	Should be left empty.
side	Either "face" or "back"
piece_side	If "board_face" a checkered board (as in American Checkers). If "board_back" a lined board (as in Turkish Checkers).
suit	Suit value (color) of board/checkers. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by piece_suit() .
angle	Angle of board in degrees. Italian Checkers rotates its board 90 degrees compared to other checkers variants.
cell_width	Width of board cell. piecepackr::game_systems() supports 1 or 2. If NULL then don't set it yet.
x, y	Cartesian coordinates (numeric vectors)
length.out	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
dfb <- checker_board(nrows = 4L, ncol = 4L)
df1 <- checker_bits(suit = "white", x = c(1, 3), y = 1)
df2 <- checker_bits(suit = "white", x = c(2, 4), y = 2)
df3 <- checker_bits(suit = "black", x = c(1, 3), y = 3)
df4 <- checker_bits(suit = "black", x = c(2, 4), y = 4)
df <- rbind(dfb, df1, df2, df3, df4)
if (require("piecepackr", quietly = TRUE) &&
    packageVersion("piecepackr") >= "1.15.0-1") {
  grid::grid.newpage()
  envir = game_systems()
  pmap_piece(df, envir = envir, default.units = "in")
}
```

chess_games_variant *Setups for chess variants*

Description

tibble data frames of setups for 6 chess variants. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
chess_chess(
  cell_width = getOption("ppdf.chess_cell_width", 1),
  ...,
  fen = "rnbqkbnr/pppppppp/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"
)

chess_chess960(
  seed = NULL,
```

```

    cell_width = getOption("ppdf.chess_cell_width", 1),
    ...,
    fen = NULL
)

chess_fischer_random_chess(
    seed = NULL,
    cell_width = getOption("ppdf.chess_cell_width", 1),
    ...,
    fen = NULL
)

chess_horde_chess(
    ...,
    cell_width = getOption("ppdf.chess_cell_width", 1),
    fen = "rnbqkbnr/pppppppp/8/1PP2PP1/PPPPPPPP/PPPPPPPP/PPPPPPPP w kq - 0 1"
)

chess_international_chess(
    cell_width = getOption("ppdf.chess_cell_width", 1),
    ...,
    fen = "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"
)

chess_jeson_mor(
    cell_width = getOption("ppdf.chess_cell_width", 1),
    ...,
    fen = "nnnnnnnnn/9/9/9/9/9/9/9/NNNNNNNN"
)

chess_monochrome_chess(
    cell_width = getOption("ppdf.chess_cell_width", 1),
    ...,
    fen = "RNBQKBNR/PPPPPPPP/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"
)

chess_racing_kings(
    cell_width = getOption("ppdf.chess_cell_width", 1),
    ...,
    fen = "8/8/8/8/8/8/krbnNBRK/qrbNBRQ w - - 0 1"
)

```

Arguments

cell_width	Width of board cell. Most renderers support 1 or 2.
...	Should be left empty.
fen	A FEN string specifying the piece placement. Only the piece placement portion (the first field) of the FEN string is used.

seed Seed that determines setup, either an integer or NULL

Details

Here are links for more information about the various games:

Game	URL
(International) Chess	https://en.wikipedia.org/wiki/Chess
Fischer Random Chess AKA Chess960	https://www.chessvariants.com/diffsetup.dir/fischer.html
Horde Chess	https://lichess.org/variant/horde
Jeson Mor	https://en.wikipedia.org/wiki/Jeson_Mor
Monochrome Chess	https://wunderland.com/WTS/Andy/Games/monochess.html
Racing Kings	https://www.chessvariants.com/diffobjective.dir/racing.html

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- chess_international_chess()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

chess_pieces

Generate chess boards and pieces

Description

`chess_board()` returns a square checkered board of desired size and type. `chess_bits()` returns chess pieces.

Usage

```
chess_board(
  nrows = 8L,
  ncols = nrows,
  x0 = 1,
  y0 = 1,
  ...,
  side = "face",
  piece_side = paste0("board_", side),
  suit = "green",
  angle = 0,
  cell_width = getOption("ppdf.chess_cell_width", 1)
)
```

```
chess_bits(
  ...,
  suit = rep(1:6, each = 6L),
  rank = rep(1:6, 6L),
  x = rep(1:6, 6L),
  y = rep(1:6, each = 6L),
  angle = 0,
  cell_width = getOption("ppdf.chess_cell_width", 1),
  length.out = NA_integer_
)
```

Arguments

nrows	Number of rows in game board
ncols	Number of columns in game board
x0	X coordinate for the center of the first cell
y0	Y coordinate for the center of the first cell
...	Should be left empty.
side	Either "face" or "back"
piece_side	If "board_face" a checkered board (as in American Checkers). If "board_back" a lined board (as in Turkish Checkers).
suit	Suit value (color) of board and pieces. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by piece_suit() .
angle	Rotation of bit/board (numeric vector of degrees, counter-clockwise). Will be coerced by piece_angle() .
cell_width	Width of board cell. piecepackr::game_systems() supports 1 or 2.
rank	Rank value of chess pieces from 1L (pawn) to 6L (king). Will be coerced by chess_rank() .
x, y	Cartesian coordinates (numeric vectors)
length.out	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
# David Silverman's "Mini chess 4x4"
dfb <- chess_board(nrows = 4L, ncol = 4L)
df1 <- chess_bits(suit = "white", rank = c("R", "Q", "K", "R"),
  x = 1:4, y= 1)
df2 <- chess_bits(suit = "white", rank = "P", x = 1:4, y= 2)
df3 <- chess_bits(suit = "black", rank = "P", x = 1:4, y= 3)
df4 <- chess_bits(suit = "black", rank = c("r", "q", "k", "r"),
  x = 1:4, y= 4)
df <- rbind(dfb, df1, df2, df3, df4)
if (require("piecepackr", quietly = TRUE) &&
  packageVersion("piecepackr") >= "1.15.0-1" &&
  requireNamespace("systemfonts", quietly = TRUE) &&
  piecepackr::has_font("Dejavu Sans") &&
  piecepackr::device_supports_unicode()) {
  grid::grid.newpage()
  envir = game_systems("dejavu")
  pmap_piece(df, envir = envir, default.units = "in")
}
```

cube_pieces

Generate cubes

Description

`cube_bits()` returns cubes.

Usage

```
cube_bits(
  ...,
  suit = 1:6,
  rank = 2L,
  x = as.double(1:6),
  y = 1,
  angle = 0,
  length.out = NA_integer_
)
```

Arguments

...	Should be left empty.
suit	Suit value (color) of cubes. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by piece_suit() .
rank	Determines the size of the cube from 8mm (rank 1) to 25mm (rank 6). Default is 10mm cubes.
x, y	Cartesian coordinates (numeric vectors)
angle	Rotation of piece (numeric vector of degrees, counter-clockwise). Will be coerced by piece_angle() .
length.out	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A [tibble::tibble\(\)](#) data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with [piecepackr::render_piece\(\)](#) and likely [ppcli::cat_piece\(\)](#).

Examples

```
df <- cube_bits(suit = 6:1, rank = 6:1, x = 6:1, y = 1)
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df)
}
if (require("piecepackr", quietly = TRUE) &&
```

```
    packageVersion("piecepackr") >= "1.16.0-3") {  
  grid::grid.newpage()  
  envir = game_systems()  
  pmap_piece(df, envir = envir, default.units = "in", op_scale = 0.5)  
}
```

dice_pieces

Generate dice.

Description

`dice_dice()` generates data frames for standard six-sided dice. `d4_dice()`, `numeral_dice()`, `d8_dice()`, `d10_dice()`, `percentile_dice()`, `d12_dice()`, and `d20_dice()` generates data frames for the classic seven polyhedral dice. `fudge_dice()` generates data frames for fudge dice.

Usage

```
dice_dice(  
  ...,  
  suit = 1:6,  
  rank = 1L,  
  x = as.double(1:6),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_  
)
```

```
d4_dice(  
  ...,  
  suit = 1:6,  
  rank = 1L,  
  x = as.double(1:6),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_  
)
```

```
fudge_dice(  
  ...,  
  suit = 1:6,  
  rank = 1L,  
  x = as.double(1:6),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_  
)
```

```
numeral_dice(  
  ...,  
  suit = 1:6,  
  rank = 1L,  
  x = as.double(1:6),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_  
)
```

```
d8_dice(  
  ...,  
  suit = 1:6,  
  rank = 1L,  
  x = as.double(1:6),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_  
)
```

```
d10_dice(  
  ...,  
  suit = 1:6,  
  rank = 1L,  
  x = as.double(1:6),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_  
)
```

```
percentile_dice(  
  ...,  
  suit = 1:6,  
  rank = 1L,  
  x = as.double(1:6),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_  
)
```

```
d12_dice(  
  ...,  
  suit = 1:6,  
  rank = 1L,  
  x = as.double(1:6),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_  
)
```

```

)

d20_dice(
  ...,
  suit = 1:6,
  rank = 1L,
  x = as.double(1:6),
  y = 1,
  angle = 0,
  length.out = NA_integer_
)

```

Arguments

...	Should be left empty.
suit	Suit value (color) of dice. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by piece_suit() .
rank	Rank value of dice from 1L (one) on up. d10 zeroes are 10L. Will be coerced by dice_rank() except percentile_dice() will coerce with percentile_dice_rank() and fudge_dice() will coerce with fudge_dice_rank() .
x, y	Cartesian coordinates (numeric vectors)
angle	Rotation of dice (numeric vector of degrees, counter-clockwise). Will be coerced by piece_angle() .
length.out	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A [tibble::tibble\(\)](#) data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with [piecepackr::render_piece\(\)](#) and likely [ppcli::cat_piece\(\)](#).

Examples

```

dfd <- dice_dice(suit = 1:6, rank = 1:6, x = 1:6, y = 1)
dfn <- numeral_dice(suit = 1:6, rank = 1:6, x = 1:6, y = 2)
dff <- fudge_dice(suit = 1:6, rank = rep(c("-", " ", "+"), 2L),
  x = 1:6, y = 3)
dfd12 <- d12_dice(suit = 1:6, rank = 1:6, x = 1:6, y = 4)

```

```
df <- rbind(dfd, dfn, dff, dfd12)
if (require("piecepackr", quietly = TRUE) &&
    packageVersion("piecepackr") >= "1.15.0-1" &&
    piecepackr:::device_supports_unicode()) {
  grid::grid.newpage()
  envir = game_systems()
  pmap_piece(df, envir = envir, default.units = "in", op_scale = 0.01)
}
```

domino_games_variant *Setups for dominoes variants*

Description

tibble data frames of setups for 10 dominoes variants. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
domino_concentration(seed = NULL)

domino_bee_donimoes(n = 7, seed = NULL)

domino_finder(seed = NULL)

domino_freecell(n = 7, seed = NULL)

domino_fujisan(seed = NULL, ..., pips = NULL, pawns = "S12M/A12C")

domino_luzon(seed = NULL)

domino_patience(seed = NULL)

domino_runners(seed = NULL)

domino_the_jubilee(seed = NULL)

domino_the_sultan(seed = NULL)
```

Arguments

seed	Seed that determines setup, either an integer or NULL
n	The number of doubles in a set e.g. n = 7 for double-6 set, n = 10 for a double-9 set, n = 13 for a double-12 set.
...	Should be left empty.
pips	A vector or matrix of Fuji-san pip layout, or NULL for a random layout. Can also be a string of integers with the rows separated by a /. The first row is the top row (y=2) and the second is the bottom row (y=1).

pawns A FEN-like string of the initial pawn placement. The four pawns S, M, C, and A are represented by their letter and empty spaces by a number.

Details

Here are links for more information about the various games:

Game	URL
Bee Donimoes	https://donkirkby.github.io/donimoes/rules.html#bee-donimoes
Concentration	http://www.domino-play.com/Games/Concentration.htm
Domino Finder	https://donkirkby.github.io/donimoes/rules.html#domino-finder
Dominoes Freecell	https://www.pagat.com/invented/domfreecell.html
(Domino) Fuji-san	http://donkirkby.github.io/donimoes/rules.html#fujisan
Luzon	http://www.domino-play.com/Games/Luzon.htm
(Domino) Patience	http://www.domino-play.com/Games/Patience.htm
Domino Runners	https://donkirkby.github.io/donimoes/rules.html#domino-runners
The Jubilee	http://www.domino-play.com/Games/Jubilee.htm
The Sultan	http://www.domino-play.com/Games/Sultan.htm

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- domino_the_jubilee()
if (require("piecepackr", quietly = TRUE) &&
    packageVersion("piecepackr") >= "1.15.0-9") {
  grid::grid.newpage()
  envir = game_systems(round = TRUE)
  pmap_piece(df, envir = envir, default.units = "in",
             trans = op_transform, op_scale = 0.5)
}
```

domino_tiles

*Generate dominoes tiles***Description**

domino_tiles() generates a data frame of dominoes tiles. By default will generate all the dominoes tiles of a double-6 set.

Usage

```
domino_tiles(
  n = 7,
  ...,
  side = "face",
  piece_side = paste0("tile_", side),
  suit = sequence(n:1, from = 1:n),
  rank = rep.int(1:n, n:1),
  cfg = "dominoes",
  x = sequence(n:1, from = 1:n),
  y = 2 * rep.int(n:1, n:1) - 0.5,
  angle = 0
)
```

Arguments

n	The number of doubles in a set e.g. n = 7 for double-6 set, n = 10 for a double-9 set, n = 13 for a double-12 set.
...	Should be left empty.
side	Either "face" or "back".
piece_side	Either "tile_face" or "tile_back".
suit	"Bottom" half of domino. Will be coerced by domino_suit() .
rank	"Top" half of domino. Will be coerced by piece_rank() .
cfg	"dominoes" or perhaps "dominoes_black", "dominoes_blue", "dominoes_green", "dominoes_red", "dominoes_white", or "dominoes_yellow".
x, y	Cartesian coordinates (numeric vectors)
angle	Rotation of dominoes (numeric vector of degrees, counter-clockwise). Will be coerced by piece_angle() .

Value

A [tibble::tibble\(\)](#) data frame with the following columns:

- "piece_side"
- "suit"

- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df_double6 <- domino_tiles()
nrow(df_double6)
df_double9 <- domino_tiles(n = 10)
nrow(df_double9)
df_double12 <- domino_tiles(n = 13)
nrow(df_double12)
df_double15 <- domino_tiles(n = 16)
nrow(df_double15)
```

games_info

Data frames with game info

Description

`chess_games()`, `domino_games()`, `checker_games()`, `piecepack_games()`, and `stackpack_games()` contain information about the games whose setups are provided by this package.

Usage

```
alquerque_games()

chess_games()

checker_games()

domino_games()

go_games()

marble_games()

morris_games()

piecepack_games()

reversi_games()
```

```
stackpack_games()
```

```
tarot_games()
```

Value

A `tibble::tibble()` data frame with character columns "game", "methods", "comment", and "url" for game name, setup function name(s), possible comment, and url for more information.

Examples

```
head(piecepack_games())
# Number of games for each game system
nrow(alquerque_games())
nrow(chess_games())
nrow(checker_games())
nrow(domino_games())
nrow(go_games())
nrow(marble_games())
nrow(morris_games())
nrow(piecepack_games())
nrow(stackpack_games())
```

go_games_variant	<i>Setups for go variants</i>
------------------	-------------------------------

Description

tibble data frames of setups for 4 go variants. Data frame output can usually be plotted with `mpmap_piece(df, default.units = "in")`.

Usage

```
go_connect6(nrows = 19L, ncols = nrows)
```

```
go_go(nrows = 19L, ncols = nrows)
```

```
go_gomoku(nrows = 15L, ncols = nrows)
```

```
go_renju(nrows = 15L, ncols = nrows)
```

Arguments

nrows	Number of rows (lines) in game board
ncols	Number of columns (lines) in game board

Details

Here are links for more information about the various games:

Game	URL
Connect6	https://en.wikipedia.org/wiki/Connect6
Go	https://en.wikipedia.org/wiki/Go_(game)
Gomoku	https://en.wikipedia.org/wiki/Gomoku
Renju	https://en.wikipedia.org/wiki/Renju

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- go_go()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.2.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

go_pieces

Generate go boards and pieces

Description

`go_board()` returns an go board. `go_bits()` returns go pieces.

Usage

```
go_board(nrows = 19L, ncols = nrows, x0 = 1, y0 = 1, ..., suit = "black")

go_bits(
  ...,
  suit = 1:6,
  x = as.double(1:6),
  y = 1,
  angle = 0,
  length.out = NA_integer_
)
```

Arguments

nrows	Number of rows in game board
ncols	Number of columns in game board
x0	X coordinate for the center of the first point
y0	Y coordinate for the center of the first point
...	Should be left empty.
suit	Suit value (color) of board/bit. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by piece_suit() .
x, y	Cartesian coordinates (numeric vectors)
angle	Rotation of piece (numeric vector of degrees, counter-clockwise). Will be coerced by piece_angle() .
length.out	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A [tibble::tibble\(\)](#) data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with [piecepackr::render_piece\(\)](#) and likely [ppcli::cat_piece\(\)](#).

Examples

```

dfb <- go_board()
dfw <- go_bits(suit = "white", x = c(1:9, 11:19), y = c(1:9, 11:19))
dfk <- go_bits(suit = "black", x = 1:19, y = 19:1)
df <- rbind(dfb, dfw, dfk)
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.2.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
dfb <- go_board(4L, 4L)
dfw <- go_bits(suit = "white", x = 1:4, y = 1:4)
dfk <- go_bits(suit = "black", x = 1:4, y = 4:1)
df <- rbind(dfb, dfw, dfk)
if (require("piecepackr", quietly = TRUE) &&
    packageVersion("piecepackr") >= "1.15.0-9") {
  grid::grid.newpage()
  envir = game_systems(background_color = "burlywood")
  pmap_piece(df, envir = envir, default.units = "in")
}

```

marble_pieces

*Generate marbles boards and pieces***Description**

marble_board() returns a square holed board of desired size and color. marble_bits() returns marble pieces.

Usage

```

marble_board(
  nrows = 4L,
  ncols = nrows,
  x0 = 1 * scale_factor,
  y0 = 1 * scale_factor,
  ...,
  suit = "cyan",
  scale_factor = 0.5
)

marble_bits(
  ...,
  suit = 1:6,
  rank = 9L,
  x = as.double(1:6),
  y = 1,
  angle = 0,
  length.out = NA_integer_
)

```

Arguments

<code>nrows</code>	Number of rows in game board (i.e. number of holes vertically)
<code>ncols</code>	Number of columns in game board (i.e. number of holes horizontally)
<code>x0</code>	X coordinate for the center of the first hole
<code>y0</code>	Y coordinate for the center of the first hole
<code>...</code>	Should be left empty.
<code>suit</code>	Suit value (color) of board and bits. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by <code>piece_suit()</code> .
<code>scale_factor</code>	Used by {ppn}. Should usually be 0.5 for square pyramidal games but could be 1 if not playing a pyramidal game.
<code>rank</code>	Determines the size of the marble from 0.5 inch diameter (rank 1) to 1.0 inch diameter (rank 9). The default is 1.0 inch diameter marbles which match the holed boards.
<code>x, y</code>	Cartesian coordinates (numeric vectors)
<code>angle</code>	Rotation of piece (numeric vector of degrees, counter-clockwise). Will be coerced by <code>piece_angle()</code> .
<code>length.out</code>	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
dfb <- marble_board(nrows = 2L, scale_factor = 1.0)
dfm <- marble_bits(x = c(1:2, 1:2, 1.5),
                  y = c(1, 1, 2, 2, 1.5),
                  suit = c(1:3, 5:6))
df <- rbind(dfb, dfm)
if (require("piecepackr", quietly = TRUE) &&
    packageVersion("piecepackr") >= "1.15.0-1") {
  grid::grid.newpage()
  envir = game_systems()
  pmap_piece(df, envir = envir, default.units = "in")
}
```

meeple_pieces *Generate meeples*

Description

`meeple_bits()` returns meeples.

Usage

```
meeple_bits(
  ...,
  side = "face",
  piece_side = paste0("bit_", side),
  suit = 1:6,
  x = as.double(1:6),
  y = 1,
  angle = 0,
  length.out = NA_integer_
)
```

Arguments

<code>...</code>	Should be left empty.
<code>side</code>	Either "face" or "back".
<code>piece_side</code>	Either "tile_face" or "tile_back".
<code>suit</code>	Suit value (color) of meeples. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by <code>piece_suit()</code> .
<code>x, y</code>	Cartesian coordinates (numeric vectors)
<code>angle</code>	Rotation of piece (numeric vector of degrees, counter-clockwise). Will be coerced by <code>piece_angle()</code> .
<code>length.out</code>	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
a <- c(0, 90, 180, 270, 0, 90)
df1 <- meeples_bits(side = "face", suit = 1:6, x = 1:6, y = 1, angle = a)
df2 <- meeples_bits(side = "top", suit = 1:6, x = 1:6, y = 2, angle = a)
df <- rbind(df1, df2)
if (require("piecepackr", quietly = TRUE) &&
    packageVersion("piecepackr") >= "1.15.0-1") {
  grid::grid.newpage()
  envir = game_systems()
  pmap_piece(df, envir = envir, default.units = "in", op_scale = 0.5)
}
```

`morris_games_variant` *Setups for morris variants*

Description

tibble data frames of setups for 7 morris variants. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
morris_three_mens_morris()

morris_five_mens_morris()

morris_six_mens_morris()

morris_seven_mens_morris()

morris_nine_mens_morris()

morris_ten_mens_morris()

morris_lasker_morris()

morris_twelve_mens_morris()
```

Details

Here are links for more information about the various games:

Game

Three Men's Morris
Five Men's Morris
Six Men's Morris
Seven Men's Morris

URL

https://en.wikipedia.org/wiki/Three_men%27s_morris
https://en.wikipedia.org/wiki/Nine_men%27s_morris#Six_men's_morris
<https://boardgamegeek.com/boardgame/25702/six-mens-morris>
https://en.wikipedia.org/wiki/Nine_men%27s_morris#Six_men's_morris

Nine Men's Morris	https://en.wikipedia.org/wiki/Nine_men%27s_morris
Ten Men's Morris aka Lasker Morris	https://en.wikipedia.org/wiki/Nine_men%27s_morris#Lasker_morris
Twelve Men's Morris	https://en.wikipedia.org/wiki/Morabaraba

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- morris_nine_mens_morris()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.2.0-8") {
  ppcli::cat_piece(df, annotate = TRUE)
}
df <- morris_three_mens_morris()
if (requireNamespace("piecepackr", quietly = TRUE)) {
  piecepackr::render_piece(df, open_device = FALSE, annotate = TRUE)
}
```

morris_pieces

Generate morris boards and pieces

Description

`morris_board()` returns various morris boards. `morris_bits()` returns morris pieces aka merels.

Usage

```
morris_board(n = 9L, x0 = 1, y0 = 1, ..., suit = "green")
```

```
morris_bits(
  ...,
  suit = 1:6,
  x = as.double(1:6),
  y = 1,
  angle = 0,
  length.out = NA_integer_
)
```

Arguments

<code>n</code>	The number of "men" the morris board is for.
<code>x0</code>	X coordinate for the center of the first point
<code>y0</code>	Y coordinate for the center of the first point
<code>...</code>	Should be left empty.
<code>suit</code>	Suit value (color) of board/bit. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by <code>piece_suit()</code> .
<code>x, y</code>	Cartesian coordinates (numeric vectors)
<code>angle</code>	Rotation of piece (numeric vector of degrees, counter-clockwise). Will be coerced by <code>piece_angle()</code> .
<code>length.out</code>	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- morris_board(9L)
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.2.0-8") {
  ppcli::cat_piece(df, annotate = TRUE)
}
df <- morris_board(n = 3L)
if (requireNamespace("piecepackr", quietly = TRUE)) {
  piecepackr::render_piece(df, open_device = FALSE, annotate = TRUE)
}
```

normalize_name	<i>Normalize name(s)</i>
----------------	--------------------------

Description

normalize_name() normalizes (game) name(s).

Usage

```
normalize_name(x, sep = "_")
```

Arguments

x	Name to normalize.
sep	Separator character.

Details

normalize_name(x, sep = "_") is used to help create R variable names while normalize_name(x, sep = "-") is used to help create filenames and LaTeX labels.

Value

Character vector of normalized names.

Examples

```
normalize_name("Fuji-san")
normalize_name("Nine Men's Morris", sep = "-")
```

piece_angle	<i>Cast to angle doubles</i>
-------------	------------------------------

Description

These functions help make sure angle values are the right double values expected by the configurations used by `piecepackr::game_systems()` and `ppn::read_ppn()`. If the input is numeric it simply uses `as.double()` and if the input is character it tries to cast to a reasonable value.

Usage

```
piece_angle(angle)
```

Arguments

angle	A numeric or character vector of angle values.
-------	--

Value

An double vector of angle values.

Examples

```
piece_angle(c("^", "<", "v", ">"))
piece_angle(c("\U0001f06e", "\U0001f038"))
```

piece_rank	<i>Cast to rank integers</i>
------------	------------------------------

Description

These functions help make sure rank values are the right integer values expected by the configurations used by `piecepackr::game_systems()` and `ppn::read_ppn()`. If the input is numeric simply use `as.integer()` and if the input is character try to cast to a reasonable value.

Usage

```
piece_rank(rank)

chess_rank(rank)

dice_rank(rank)

fudge_dice_rank(rank)

percentile_dice_rank(rank)

tarot_rank(rank)
```

Arguments

rank A numeric or character vector of rank values.

Value

An integer vector of rank values.

Examples

```
chess_rank(c("P", "N", "B", "R", "Q", "K"))
chess_rank(c("\u265f", "\u265e", "\u265d", "\u265c", "\u265b", "\u265a"))
dice_rank(c("1", "2", "6"))
dice_rank(c("\u2680", "\u2681", "\u2685"))
fudge_dice_rank(c("-", " ", "+"))
percentile_dice_rank(c("00", "10", "90"))
piece_rank(c("n", "a", "2", "3", "4", "5"))
```

```

piece_rank(c("\U0001f06e", "\U0001f038"))
piece_rank(c("zero", "one", "eighteen"))
tarot_rank(c("ace", "ten", "jack", "knight", "queen", "king"))

```

piece_suit

Cast to suit integers

Description

These functions help make sure suit values are the right integer values expected by the configurations used by `piecepackr::game_systems()` and `ppn::read_ppn()`. If the input is numeric it simply uses `as.integer()` and if the input is character it tries to cast to a reasonable value.

Usage

```

piece_suit(suit)

domino_suit(suit)

tarot_suit(suit)

```

Arguments

`suit` A numeric or character vector of suit values.

Value

An integer vector of suit values.

Examples

```

piece_suit(c("red", "black", "green", "blue", "yellow", "white"))
piece_suit(c("R", "K", "G", "B", "Y", "W"))
piece_suit(c("suns", "moons", "crowns", "arms"))
piece_suit(c("S", "M", "C", "A"))
domino_suit(c("\U0001f06e", "\U0001f038"))
domino_suit(c("zero", "one", "eighteen"))
tarot_suit(c("\u2665", "\u2660", "\u2663", "\u2666"))
tarot_suit(c("hearts", "spades", "clubs", "diamonds", "trumps"))
tarot_suit(c("H", "S", "C", "D"))

```

piecepack_board	<i>Setup various game boards using piecepack tiles</i>
-----------------	--

Description

piecepack_rectangular_board() returns a tibble data frame of a rectangular board of desired size made using a maximum number of piecepack tiles. It will use either “cells”, “points”, and/or “rivers” as necessary. piecepack_donut_board() returns 24 tiles arranged 5x5 with hole in middle.

Usage

```
piecepack_donut_board(
  ...,
  side = "back",
  piece_side = paste0("tile_", side),
  suit = rep(1:4, each = 6L),
  rank = rep.int(1:6, 4L),
  cfg = "piecepack",
  x0 = 1,
  y0 = 1,
  angle = 0
)

piecepack_rectangular_board(
  nrows = 8L,
  ncols = 8L,
  x0 = 1,
  y0 = 1,
  ...,
  max_tiles = 24L,
  suit = rep.int(1:4, 6L),
  rank = rep(1:6, each = 4L),
  cfg = "piecepack",
  angle = 0
)
```

Arguments

...	Should be left empty.
side	Either "face" or "back".
piece_side	Either "tile_face" or "tile_back".
suit	Vector of suit values to use for tile back (will be repeated and coerced by piece_suit()).
rank	Vector of rank values to use for tile back (will be repeated and coerced by piece_rank()).

cfg	"piecepack" or perhaps "playing_cards_expansion", "dual_piecepacks_expansion", or "subpack".
x0	X coordinate for the center of the first cell/point for <code>piecepack_rectangular_board()</code> . X coordinate for center of bottom-left tile for <code>piecepack_donut_board()</code> .
y0	Y coordinate for the center of the first cell/point for <code>piecepack_rectangular_board()</code> . Y coordinate for center of bottom-left tile for <code>piecepack_donut_board()</code> .
angle	Rotation of piece (numeric vector of degrees, counter-clockwise). Will be coerced by <code>piece_angle()</code> .
nrows	Number of rows in game board
ncols	Number of columns in game board
max_tiles	Maximum number of tiles that can be used

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- piecepack_rectangular_board(4L, 4L)
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

piecepack_games_checkers

Setups for checkers variants playable with a piecepack

Description

tibble data frames of setups for 17 checkers variants playable with a piecepack. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

`piecepack_american_checkers()`

`piecepack_american_pool_checkers()`

`piecepack_bashni()`

`piecepack_brazilian_checkers()`

`piecepack_checkers()`

`piecepack_column_checkers()`

`piecepack_corner_checkers()`

`piecepack_czech_checkers()`

`piecepack_english_checkers()`

`piecepack_gothic_checkers()`

`piecepack_italian_checkers()`

`piecepack_jamaican_checkers()`

`piecepack_one_way_checkers()`

`piecepack_portuguese_checkers()`

`piecepack_russian_checkers()`

`piecepack_spanish_checkers()`

`piecepack_thai_checkers()`

`piecepack_turkish_checkers()`

`piecepack_unified_pool_checkers()`

`piecepack_zimbabwean_pool_checkers()`

Details

Here are links for more information about the various checkers variants:

Game

(American) Checkers AKA (English) Draughts
American Pool Checkers

URL

<https://www.ludism.org/ppwiki/Checkers>
<https://draughts.github.io/american-pool-checkers/>

Bashni AKA Column Checkers
 Brazilian Checkers AKA Brazilian Draughts
 Czech Checkers AKA Czech Draughts
 Corner Checkers AKA Corner Draughts
 Gothic Checkers AKA Gothic Draughts
 Italian Checkers AKA Italian Draughts
 Jamaican Checkers AKA Jamaican Draughts
 One Way Checkers AKA One Way Draughts
 Portuguese Checkers AKA Portuguese Draughts
 Russian Checkers AKA Russian Draughts
 Spanish Checkers AKA Spanish Draughts
 Thai Checkers AKA Thai Draughts AKA Mak-hot AKA Makhos
 Turkish Checkers AKA Turkish Draughts AKA Dama
 Unified Pool Checkers
 Zimbabwean Pool Checkers

<https://en.wikipedia.org/wiki/Bashni>
https://en.wikipedia.org/wiki/Brazilian_draughts
https://en.wikipedia.org/wiki/Czech_draughts
<https://brainking.com/en/GameRules?tp=30>
<http://mlwi.magix.net/bg/gothiccheckersvariant>
https://en.wikipedia.org/wiki/Italian_draughts
<https://web.archive.org/web/20230605023244/http://brainking.com/en/GameRules?tp=31>
<http://www.fpdamas.pt/regras/>
https://en.wikipedia.org/wiki/Russian_draughts
<https://mindsports.nl/index.php/on-the-evolution-of-checkers>
<https://checkers.fandom.com/wiki/Mak-hot>
https://en.wikipedia.org/wiki/Turkish_draughts
<https://wiegerw.github.io/pdn/gametype.html>
<https://wiegerw.github.io/pdn/gametype.html>

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- piecepack_american_checkers()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

piecepack_games_chess *Setups for chess variants playable with a piecepack*

Description

tibble data frames of setups for 11 chess variants playable with a piecepack. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```

piecepack_alice_chess(has_subpack = FALSE, max_tiles = 24L)

piecepack_baroque_chess(has_subpack = FALSE)

piecepack_chaturaji(has_subpack = FALSE)

piecepack_chess(has_subpack = FALSE)

piecepack_chess960(..., seed = NULL, has_subpack = FALSE)

piecepack_fischer_random_chess(..., seed = NULL, has_subpack = FALSE)

piecepack_chinese_chess(has_subpack = FALSE)

piecepack_four_seasons_chess(has_subpack = FALSE)

piecepack_international_chess(has_subpack = FALSE)

piecepack_jeson_mor()

piecepack_japanese_chess(has_subpack = FALSE, cfg2 = "piecepack")

piecepack_minishogi()

piecepack_racing_kings()

piecepack_shogi(has_subpack = FALSE, cfg2 = "piecepack")

piecepack_ultima(has_subpack = FALSE)

piecepack_xiangqi(has_subpack = FALSE)

```

Arguments

<code>has_subpack</code>	Has a piecepack subpack
<code>max_tiles</code>	Maximum number of (piecepack) tiles available to build boards
<code>...</code>	Should be left empty.
<code>seed</code>	Seed that determines setup, either an integer or NULL
<code>cfg2</code>	A string of a piecepack expansion (or perhaps "piecepack" for a second piecepack)

Details

Here are links for more information about the various chess variants:

Game

Alice Chess

URL

https://en.wikipedia.org/wiki/Alice_chess

Chaturaji	https://www.ludism.org/ppwiki/Chaturaji
Fischer Random Chess AKA Chess960	https://www.chessvariants.com/diffsetup.dir/fischer.html
Four Seasons Chess	https://www.chessvariants.com/historic.dir/4seiz.html
(International) Chess	https://www.ludism.org/ppwiki/Chess
Jeson Mor	https://en.wikipedia.org/wiki/Jeson_Mor
Minishogi	https://en.wikipedia.org/wiki/Minishogi
Racing Kings	https://www.chessvariants.com/diffobjective.dir/racing.html
Shogi AKA Japanese Chess	https://www.ludism.org/ppwiki/Shogi
Ultima AKA Baroque Chess	https://en.wikipedia.org/wiki/Baroque_chess
Xiangqi AKA Chinese Chess	https://www.ludism.org/ppwiki/Xiangqi

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- piecepack_chess()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

piecepack_games_original

Setups for various original piecepack games

Description

tibble data frames of setups for 53 games playable with a piecepack. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
piecepack_alien_city(seed = NULL, tiles = NULL)

piecepack_black_pawn_trucking(seed = NULL)

piecepack_brain_burn(seed = NULL)

piecepack_burbuja(seed = NULL, tiles = NULL)

piecepack_cardinals_guards(seed = NULL, tiles = NULL)

piecepack_cell_management(seed = NULL)

piecepack_chariots()

piecepack_chinese_checkers()

piecepack_halma()

piecepack_coin_collectors(seed = NULL, tiles = NULL, coins = NULL, dice = NULL)

piecepack_climbing_man(seed = NULL, variant = c("Basic", "Free"))

piecepack_crocodile_hop(seed = NULL)

piecepack_desfases(seed = NULL, tiles = NULL, dice = NULL)

piecepack_dominoids(..., seed = NULL)

piecepack_easy_slider(seed = NULL, tiles = NULL, coins = NULL, pawns = NULL)

piecepack_everest()

piecepack_four_blind_mice()

piecepack_froggy_bottom()

piecepack_fujisan(seed = NULL, coins = NULL, dice = NULL, pawns = "S12M/A12C")

piecepack_galaxy_express(seed = NULL)

piecepack_iceberg(seed = NULL, tiles = NULL)

piecepack_ice_floe()

piecepack_japan(seed = NULL)

piecepack_klondike(seed = NULL)
```

```
piecepack_lab_rats(seed = NULL)

piecepack_landlocked(seed = NULL)

piecepack_ley_lines()

piecepack_mathrix(seed = NULL, coins = NULL)

piecepack_pawns_crossing(..., seed = NULL, n_players = 2L)

piecepack_piecepackman(seed = NULL, variant = "Roundabout")

piecepack_one_man_thrag(seed = NULL)

piecepack_pass_the_food()

piecepack_piece_gaps(seed = NULL)

piecepack_piece_packing_pirates(seed = NULL)

piecepack_plans_of_action(seed = NULL, coins = NULL)

piecepack_relativity(seed = NULL, coins = NULL)

piecepack_san_andreas()

piecepack_sarcophagus(seed = NULL)

piecepack_ship_it(..., seed = NULL)

piecepack_shopping_mall(seed = NULL, cfg2 = "go")

piecepack_skyscrapers(seed = NULL, tiles = NULL)

piecepack_slides_of_action()

piecepack_speedy_towers(n_players = 2, seed = NULL)

piecepack_steppin_stones(seed = NULL)

piecepack_tariffs(
  variant = c("simple zones", "grouped zones", "scattered zones")
)

piecepack_the_in_crowd()

piecepack_the_magic_bag(seed = NULL)
```

```

piecepack_the_penguin_game(seed = NULL)

piecepack_tornado_vs_trailer_park()

piecepack_tower_of_babel(seed = NULL, tiles = NULL)

piecepack_accordion(seed = NULL, tiles = NULL)

piecepack_tracers()

piecepack_triactor(seed = NULL, cfg2 = "playing_cards_expansion")

piecepack_tula(
  seed = NULL,
  tiles = NULL,
  variant = c("Original", "Variant 1", "Variant 2", "Variant 3", "Variant 4")
)

piecepack_wormholes()

```

Arguments

seed	Seed that determines setup, either an integer or NULL
tiles	String of tile layout
coins	String of coin layout
dice	String of dice layout
variant	Name of variant
...	Should be left empty.
pawns	A FEN-like string of the initial pawn placement. The four pawns S, M, C, and A are represented by their letter and empty spaces by a number.
n_players	Number of players
cfg2	A string of a piecepack expansion (or perhaps "piecepack" for a second piecepack)

Details

Here are links for more information about the various games:

Game	URL
Alien City	https://www.ludism.org/ppwiki/AlienCity
Black Pawn Trucking	https://www.ludism.org/ppwiki/BlackPawnTrucking
Brain Burn	https://www.ludism.org/ppwiki/BrainBurn
Burbuja	https://www.ludism.org/ppwiki/Burbuja
Cardinal's Guards	https://www.ludism.org/ppwiki/CardinalsGuards
Cell Management	https://www.ludism.org/ppwiki/CellManagement
Chariots	https://www.ludism.org/ppwiki/Chariots

Chinese Checkers AKA (Piecepack) Halma	https://www.ludism.org/ppwiki/ChineseCheckers
Climbing Man	https://www.ludism.org/ppwiki/ClimbingMan
Coin Collectors	https://www.ludism.org/ppwiki/CoinCollectors
Crocodile Hop	https://www.ludism.org/ppwiki/CrocodileHop
Desfases	https://www.ludism.org/ppwiki/Desfases
Dominoids	https://www.ludism.org/ppwiki/Dominoids
Easy Slider	https://www.ludism.org/ppwiki/EasySlider
Everest	https://www.ludism.org/ppwiki/Everest
Four Blind Mice	https://www.ludism.org/ppwiki/FourBlindMice
Froggy Bottom	https://www.ludism.org/ppwiki/FroggyBottom
Fuji-san	https://www.ludism.org/ppwiki/Fuji-san
Galaxy Express	https://www.ludism.org/ppwiki/GalaxyExpress
Ice Floe	https://www.ludism.org/ppwiki/IceFloe
Iceberg	https://www.ludism.org/ppwiki/Iceberg
Japan	https://www.ludism.org/ppwiki/Japan
Lab Rats	https://www.ludism.org/ppwiki/LabRats
Landlocked	https://www.ludism.org/ppwiki/Landlocked
Ley Lines	https://www.ludism.org/ppwiki/LeyLines
Mathrix	https://www.ludism.org/ppwiki/Mathrix
One Man Thrag!	https://www.ludism.org/ppwiki/OneManThrag
Pass the Food	https://www.ludism.org/ppwiki/PassTheFood
Pawns Crossing	https://www.ludism.org/ppwiki/PawnsCrossing
Piece Gaps	https://www.ludism.org/ppwiki/PieceGaps
Piece Packing Pirates	https://www.ludism.org/ppwiki/PiecePackingPirates
(Piecepack) Klondike	https://www.ludism.org/ppwiki/PiecepackKlondike
Piecepackman	https://www.ludism.org/ppwiki/Piecepackman
Plans of Action	https://www.ludism.org/ppwiki/PlansOfAction
Relativity	https://www.ludism.org/ppwiki/Relativity
San Andreas	https://www.ludism.org/ppwiki/SanAndreas
Sarcophagus	https://www.ludism.org/ppwiki/Sarcophagus
Ship It!	https://www.ludism.org/ppwiki/Ship_It
Shopping Mall	https://www.ludism.org/ppwiki/ShoppingMall
Skyscrapers	https://www.ludism.org/ppwiki/Skyscrapers
Slides of Action	https://www.ludism.org/ppwiki/SlidesOfAction
Speedy Towers	https://www.ludism.org/ppwiki/SpeedyTowers
Steppin' Stones	https://www.ludism.org/ppwiki/Steppin'_Stones
Tariffs	https://www.ludism.org/ppwiki/Tariffs
The "In" Crowd	https://www.ludism.org/ppwiki/TheInCrowd
The Magic Bag	https://www.ludism.org/ppwiki/The_Magic_Bag
The Penguin Game	https://www.ludism.org/ppwiki/PenguinGame
Tornado vs. Trailer Park	https://www.ludism.org/ppwiki/Tornado_vs._Trailer_Park
Tower of Babel AKA (Piecepack) Accordion	https://www.ludism.org/ppwiki/TowerOfBabel
Tracers	https://www.ludism.org/ppwiki/Tracers
Triactor	https://www.ludism.org/ppwiki/Triactor
Tula	https://www.ludism.org/ppwiki/Tula
Wormholes	https://www.ludism.org/ppwiki/Wormholes

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- piecepack_relativity(seed = 42)
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

piecepack_games_other *Setups for other modern games playable with a piecepack*

Description

tibble data frames of setups for 15 other games playable with a piecepack. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
piecepack_12345ive()

piecepack_breakthrough()

piecepack_change_change(seed = NULL, coins = NULL)

piecepack_crossings()

piecepack_dao()

piecepack_dodgem(nrows = 6L, variant = "vout")

piecepack_evade()

piecepack_kangaroo()
```

```
piecepack_kings_valley(variant = "standard")
```

```
piecepack_leapfrog()
```

```
piecepack_grasshopper()
```

```
piecepack_lines_of_action()
```

```
piecepack_lukawan()
```

```
piecepack_quatri()
```

```
piecepack_three_musketeers()
```

Arguments

seed	Seed that determines setup, either an integer or NULL
coins	String of coin layout
nrows	Number of rows (and columns) in game board.
variant	Name of variant.

Details

Here are links for more information about the various other games:

Game	URL
12345ive!	https://boardgamegeek.com/boardgame/154644/12345ive
Breakthrough	https://en.wikipedia.org/wiki/Breakthrough_(board_game)
Crossings	https://en.wikipedia.org/wiki/Crossings_(game)
Change Change	https://www.ludism.org/ppwiki/ChangeChange
Dao	https://boardgamegeek.com/boardgame/948/dao
Dodgem	https://en.wikipedia.org/wiki/Dodgem
Grasshopper	http://www.cynningstan.com/game/71/grasshopper
Evade	https://www.ludism.org/ppwiki/Evade
Kangaroo	https://boardgamegeek.com/boardgame/6132/kangaroo-the-jumping-game
King's Valley	https://www.logygames.com/english/kingsvalley.html
Leapfrog	https://youtu.be/LWuVDFmPkZs
Lines of Action	https://en.wikipedia.org/wiki/Lines_of_Action
Lukawan	https://ludism.org/ppwiki/Lukawan
Quatri	https://www.ludism.org/ppwiki/Quatri
Three Musketeers	https://boardgamegeek.com/boardgame/21861/three-musketeers

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"

- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- piecepack_dao()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

piecepack_games_traditional

Setups for other traditional games playable with a piecepack

Description

tibble data frames of setups for 19 other traditional games playable with a piecepack. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
piecepack_achi()

piecepack_alquerque(has_matchsticks = FALSE)

piecepack_awithlagnannai_mosona(has_matchsticks = FALSE)

piecepack_backgammon()

piecepack_baghchal(has_matchsticks = FALSE)

piecepack_brandubh()

piecepack_cribbage()

piecepack_cribbage_board()

piecepack_dara()
```

```

piecepack_four_field_kono()

piecepack_julgonu()

piecepack_ludo()

piecepack_nine_mens_morris(has_matchsticks = FALSE)

piecepack_salta(has_subpack = FALSE)

piecepack_seega()

piecepack_tablan()

piecepack_tablut(die_width = 0.63)

piecepack_tic_tac_toe()

piecepack_twelve_mens_morris(has_matchsticks = FALSE)

piecepack_yote()

```

Arguments

has_matchsticks	Has matchsticks
has_subpack	Has a piecepack subpack
die_width	Width of dice

Details

Here are links for more information about the various other traditional games:

Game	URL
Achi	https://en.wikipedia.org/wiki/Achi_(game)
Alquerque AKA Qirkat	https://en.wikipedia.org/wiki/Alquerque
Awithlakkannai Mosona	https://en.wikipedia.org/wiki/Awithlakkannai_Mosona
Backgammon	https://www.ludism.org/ppwiki/Backgammon
Bagh-chal	https://www.ludism.org/ppwiki/Bagh-Chal
Brandubh AKA Brandub	http://www.cyclingstan.com/game/125/brandub
Cribbage	https://www.ludism.org/ppwiki/Cribbage
Dara	http://www.cyclingstan.com/game/148/dara
Four Field Kono	https://www.ludism.org/ppwiki/FourFieldKono
Jul-Gonu	https://www.ludism.org/ppwiki/JulGonu
Ludo	https://en.wikipedia.org/wiki/Ludo_(board_game)
Nine Men's Morris	https://en.wikipedia.org/wiki/Nine_men%27s_morris
Salta	https://en.wikipedia.org/wiki/Salta_(game)
Seega	http://www.cyclingstan.com/game/120/seega

Tablan	http://www.cyclingstan.com/game/229/tablan
Tablut	https://www.ludism.org/ppwiki/Tablut
Tic-tac-toe	https://en.wikipedia.org/wiki/Tic-tac-toe
Twelve Men's Morris	http://www.cyclingstan.com/game/111/twelve-mens-morris
Yoté	https://en.wikipedia.org/wiki/Yot%C3%A9

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- piecepack_four_field_kono()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

piecepack_pieces	<i>Generate piecepack pieces</i>
------------------	----------------------------------

Description

`piecepack_coins()` generates a data frame of piecepack coins. `piecepack_dice()` generates a data frame of piecepack dice. `piecepack_matchsticks()` generates a data frame of piecepack matchsticks. `piecepack_pawns()` generates a data frame of piecepack pawns. `piecepack_pyramids()` generates a data frame of piecepack pyramids. `piecepack_saucers()` generates a data frame of piecepack saucers. `piecepack_tiles()` generates a data frame of piecepack tiles.

Usage

```
piecepack_coins(  
  ...,  
  side = "face",  
  piece_side = paste0("coin_", side),  
  suit = rep(1:4, each = 6L),  
  rank = rep.int(1:6, 4L),  
  cfg = "piecepack",  
  x = as.double(rep.int(1:6, 4L)),  
  y = as.double(rep(4:1, each = 6L)),  
  angle = 0,  
  length.out = NA_integer_  
)  
  
piecepack_dice(  
  ...,  
  suit = 1:4,  
  rank = 1L,  
  cfg = "piecepack",  
  x = as.double(1:4),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_  
)  
  
piecepack_matchsticks(  
  ...,  
  side = "face",  
  piece_side = paste0("matchstick_", side),  
  suit = rep(1:4, each = 6L),  
  rank = rep.int(1:6, 4L),  
  cfg = "piecepack",  
  x = as.double(rep.int(1:6, 4L)),  
  y = 3 * rep(4:1, each = 6L),  
  angle = 0,  
  length.out = NA_integer_  
)  
  
piecepack_pawns(  
  ...,  
  side = "face",  
  piece_side = paste0("pawn_", side),  
  suit = 1:4,  
  cfg = "piecepack",  
  x = as.double(1:4),  
  y = 1,  
  angle = 0,  
  length.out = NA_integer_
```

```

)

piecepack_pyramids(
  ...,
  side = "top",
  piece_side = paste0("pyramid_", side),
  suit = rep(1:4, each = 6L),
  rank = rep.int(1:6, 4L),
  cfg = "piecepack",
  x = as.double(rep.int(1:6, 4L)),
  y = as.double(rep(4:1, each = 6L)),
  angle = 0,
  length.out = NA_integer_
)

piecepack_saucers(
  ...,
  side = "face",
  piece_side = paste0("saucer_", side),
  suit = 1:4,
  cfg = "piecepack",
  x = as.double(1:4),
  y = 1,
  angle = 0,
  length.out = NA_integer_
)

piecepack_tiles(
  ...,
  side = "face",
  piece_side = paste0("tile_", side),
  suit = rep(1:4, each = 6L),
  rank = rep.int(1:6, 4L),
  cfg = "piecepack",
  x = 2 * rep.int(1:6, 4L),
  y = 2 * rep(4:1, each = 6L),
  angle = 0,
  length.out = NA_integer_
)

```

Arguments

...	Should be left empty.
side	Either "face" or "back".
piece_side	Either "tile_face" or "tile_back".
suit	Integer vector with values between 1L and 4L. 1L is "suns", 2L is "moons", 3L is "crowns", 4L is "arms". Will be coerced by piece_suit() .

rank	Integer vector with values between 1L (null) and 6L (five). Will be coerced by <code>piece_rank()</code> .
cfg	"piecepack" or perhaps "playing_cards_expansion", "dual_piecepacks_expansion", or "subpack".
x, y	Cartesian coordinates (numeric vectors)
angle	Rotation of piece (numeric vector of degrees, counter-clockwise). Will be coerced by <code>piece_angle()</code> .
length.out	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df_coins <- piecepack_coins()
df_dice <- piecepack_dice()
df_pawns <- piecepack_pawns()
df_pyramids <- piecepack_pyramids()
df_saucers <- piecepack_saucers()
df_tiles <- piecepack_tiles()
```

piecepack_preview	<i>Preview layout of a piecepack set</i>
-------------------	--

Description

`piecepack_preview()` returns a data frame representing a preview layout of the pieces in a standard piecepack set.

Usage

```
piecepack_preview(cfg = "piecepack")
```

Arguments

`cfg` "piecepack" or perhaps "playing_cards_expansion", "dual_piecepacks_expansion", or "subpack".

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- piecepack_preview()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-2") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

ppdf-defunct

Defunct functions

Description

These functions are defunct in this release of ppdf. Use the replacement function shown in the table below.

Usage

`chess_angle(...)`

`domino_angle(...)`

`piecepack_angle(...)`

`tarot_angle(...)`

`alquerque_suit(...)`

```
checker_suit(...)
chess_suit(...)
go_suit(...)
marble_suit(...)
morris_suit(...)
piecepack_suit(...)
domino_rank(...)
piecepack_rank(...)
checkers_american_checkers(...)
checkers_american_pool_checkers(...)
checkers_board(...)
checkers_brazilian_checkers(...)
checkers_breakthrough(...)
checkers_by_name(...)
checkers_canadian_checkers(...)
checkers_crossings(...)
checkers_czech_checkers(...)
checkers_checkers(...)
checkers_dameo(...)
checkers_english_checkers(...)
checkers_focus(...)
checkers_four_field_kono(...)
checkers_frisian_checkers(...)
checkers_gothic_checkers(...)
```

checkers_grasshopper(...)
checkers_international_checkers(...)
checkers_italian_checkers(...)
checkers_jamaican_checkers(...)
checkers_julgonu(...)
checkers_lines_of_action(...)
checkers_none(...)
checkers_portuguese_checkers(...)
checkers_russian_checkers(...)
checkers_spanish_checkers(...)
checkers_thai_checkers(...)
checkers_turkish_checkers(...)
chess_by_name(...)
dominoes_by_name(...)
dominoes_concentration(...)
dominoes_domino_finder(...)
dominoes_domino_runners(...)
dominoes_fujisan(...)
dominoes_luzon(...)
dominoes_none(...)
dominoes_patience(...)
dominoes_the_jubilee(...)
dominoes_tiles(...)
games_checkers(...)

```

games_chess(...)

games_dominoes(...)

games_piecepack(...)

games_stackpack(...)

piecepack_by_name(...)

piecepack_piecepack_accordion(...)

piecepack_piecepack_halma(...)

piecepack_piecepack_klondike(...)

piecepack_rect_board_tiles(...)

stackpack_by_name(...)

```

Arguments

... Ignored.

Details

Defunct function

```

checkers_by_name()
chess_by_name()
dominoes_by_name()
dominoes_concentration()
dominoes_domino_finder()
dominoes_domino_runners()
dominoes_fujisan()
dominoes_luzon()
dominoes_none()
dominoes_patience()
dominoes_the_jubilee()
dominoes_tiles()
games_checkers()
games_chess()
games_dominoes()
games_piecepack()
games_stackpack()
piecepack_by_name()
piecepack_piecepack_accordion()
piecepack_piecepack_halma()
piecepack_piecepack_klondike()

```

Replacement function

```

checker_setup_by_name()
chess_setup_by_name()
domino_setup_by_name()
domino_concentration()
domino_finder()
domino_runners()
domino_fujisan()
domino_luzon()
domino_none()
domino_patience()
domino_the_jubilee()
domino_tiles()
checker_games()
chess_games()
domino_games()
piecepack_games()
stackpack_games()
piecepack_setup_by_name()
piecepack_accordion()
piecepack_halma()
piecepack_klondike()

```

piecepack_rect_board_tiles()	piecepack_rectangular_board()
stackpack_by_name()	stackpack_setup_by_name()

reversi_games_variant *Setups for reversi variants*

Description

tibble data frames of setups for 2 reversi variants. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
reversi_ming_mang(nrows = 8L, ncols = nrows)
```

```
reversi_reversi(nrows = 8L, ncols = nrows)
```

Arguments

nrows	Number of rows in game board
ncols	Number of columns in game board

Details

Here are links for more information about the various games:

Game	URL
Ming Mang	https://boardgamegeek.com/boardgame/40573/ming-mang
Reversi	https://en.wikipedia.org/wiki/Reversi

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- reversi_reversi()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

reversi_pieces	<i>Generate reversi boards and pieces</i>
----------------	---

Description

reversi_board() returns various reversi boards. reversi_bits() returns reversi pieces.

Usage

```
reversi_board(
  nrows = 8L,
  ncols = nrows,
  x0 = 1,
  y0 = 1,
  ...,
  side = "face",
  piece_side = paste0("board_", side),
  suit = "black",
  angle = 0
)

reversi_bits(
  ...,
  side = "face",
  piece_side = paste0("bit_", side),
  suit = 1:6,
  x = as.double(1:6),
  y = 1,
  angle = 0,
  length.out = NA_integer_
)
```

Arguments

nrows	Number of rows in game board
ncols	Number of columns in game board
x0	X coordinate for the center of the first point
y0	Y coordinate for the center of the first point
...	Should be left empty.

side	Either "face" or "back"
piece_side	If "board_face" a lined board. If "board_back" a checkered board.
suit	Suit value (color) of board/bit. 1L is "red", 2L is "black", 3L is "green", 4L is "blue", 5L is "yellow", and 6L is "white". Will be coerced by <code>piece_suit()</code> .
angle	Angle of board in degrees.
x, y	Cartesian coordinates (numeric vectors)
length.out	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
dfb <- reversi_board()
dfp <- reversi_bits(suit = c("black", "white"), x = c(4, 5), y = c(4, 4))
df <- rbind(dfb, dfp)
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

setup_by_name	<i>Setups by game name</i>
---------------	----------------------------

Description

`setup_by_name()` returns setup data frames by game name and game system. `alquerque_setup_by_name()`, `checker_setup_by_name()`, `chess_setup_by_name()`, `dominoes_setup_by_name()`, `marble_setup_by_name()`, `piecepack_setup_by_name()` and `stackpack_setup_by_name()` are aliases that set the game system.

Usage

```
setup_by_name(  
  name,  
  system = known_game_systems,  
  ...,  
  getter = function(x) get(x, envir = getNamespace("ppdf"))  
)  
  
alquerque_setup_by_name(  
  name,  
  ...,  
  getter = function(x) get(x, envir = getNamespace("ppdf"))  
)  
  
checker_setup_by_name(  
  name,  
  ...,  
  getter = function(x) get(x, envir = getNamespace("ppdf"))  
)  
  
chess_setup_by_name(  
  name,  
  ...,  
  getter = function(x) get(x, envir = getNamespace("ppdf"))  
)  
  
domino_setup_by_name(  
  name,  
  ...,  
  getter = function(x) get(x, envir = getNamespace("ppdf"))  
)  
  
go_setup_by_name(  
  name,  
  ...,  
  getter = function(x) get(x, envir = getNamespace("ppdf"))  
)  
  
marble_setup_by_name(  
  name,  
  ...,  
  getter = function(x) get(x, envir = getNamespace("ppdf"))  
)  
  
morris_setup_by_name(  
  name,  
  ...,  
  getter = function(x) get(x, envir = getNamespace("ppdf"))  
)
```

```

)

piecepack_setup_by_name(
  name,
  ...,
  getter = function(x) get(x, envir = getNamespace("ppdf"))
)

reversi_setup_by_name(
  name,
  ...,
  getter = function(x) get(x, envir = getNamespace("ppdf"))
)

stackpack_setup_by_name(
  name,
  ...,
  getter = function(x) get(x, envir = getNamespace("ppdf"))
)

tarot_setup_by_name(
  name,
  ...,
  getter = function(x) get(x, envir = getNamespace("ppdf"))
)

```

Arguments

name	Game name. Will be normalized by normalize_name() .
system	Game system.
...	Additional arguments to pass to the underlying setup function.
getter	Function with a single function name argument to use. By default will just look for functions in this package but a developer may want to consider dynGet() or a wrapper around get() with a custom <code>envir</code> argument.

Value

A [tibble::tibble\(\)](#) data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with [piecepackr::render_piece\(\)](#) and likely [ppcli::cat_piece\(\)](#).

Examples

```
df <- chess_setup_by_name("chess")
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

 setup_none

Zero-row data frames

Description

alquerque_none(), checker_none(), chess_none(), domino_none(), go_none(), marble_none(), piecepack_none(), and stackpack_none() return zero-row data frames.

Usage

```
alquerque_none()
checker_none()
chess_none()
domino_none()
go_none()
marble_none()
morris_none()
piecepack_none()
reversi_none()
stackpack_none()
tarot_none()
```

Value

A data frame with zero rows.

Examples

```
checker_none()
```

stackpack_games_other *Setups for games playable with a piecepack stackpack*

Description

tibble data frames of setups for 10 other games playable with a piecepack stackpack. Data frame output can usually be plotted with `pmap_piece(df, default.units = "in")`.

Usage

```
stackpack_alice_chess(max_tiles = 24L)

stackpack_baroque_chess()

stackpack_chaturaji()

stackpack_chess()

stackpack_chess960(..., seed = NULL)

stackpack_fischer_random_chess(..., seed = NULL)

stackpack_chinese_chess()

stackpack_four_seasons_chess()

stackpack_horde_chess()

stackpack_international_chess()

stackpack_japanese_chess()

stackpack_salta()

stackpack_shogi()

stackpack_ultima()

stackpack_xiangqi()
```

Arguments

<code>max_tiles</code>	Maximum number of (piecepack) tiles available to build boards
<code>...</code>	Should be left empty.
<code>seed</code>	Seed that determines setup, either an integer or NULL

Details

Here are links for more information about the games:

Game	URL
Alice Chess	https://en.wikipedia.org/wiki/Alice_chess
Chaturaji	https://www.ludism.org/ppwiki/Chaturaji
Fischer Random Chess AKA Chess960	https://www.chessvariants.com/diffsetup.dir/fischer.html
Four Seasons Chess	https://www.chessvariants.com/historic.dir/4seiz.html
Horde Chess	https://lichess.org/variant/horde
(International) Chess	https://www.ludism.org/ppwiki/Chess
Salta	https://en.wikipedia.org/wiki/Salta_(game)
Shogi aka Japanese Chess	https://www.ludism.org/ppwiki/Shogi
Ultima aka Baroque Chess	https://en.wikipedia.org/wiki/Baroque_chess
Xiangqi AKA Chinese Chess	https://www.ludism.org/ppwiki/Xiangqi

Value

A `tibble::tibble()` data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with `piecepackr::render_piece()` and likely `ppcli::cat_piece()`.

Examples

```
df <- stackpack_chess()
if (requireNamespace("ppcli", quietly = TRUE) &&
    packageVersion("ppcli") >= "0.3.0-1") {
  ppcli::cat_piece(df, annotate = TRUE)
}
```

tarot_pieces

Generate tarot cards

Description

`tarot_cards()` returns tarot cards.

Usage

```
tarot_cards(
  ...,
  side = "face",
  piece_side = paste0("card_", side),
  suit = rep(1:4, each = 13L),
  rank = rep.int(c(1:11, 13:14), 4L),
  x = 2.5 * rep.int(1:13, 4L),
  y = 3.5 * rep(4:1, each = 13L),
  angle = 0,
  length.out = NA_integer_
)
```

Arguments

...	Should be left empty.
side	Either "face" or "back".
piece_side	Either "tile_face" or "tile_back".
suit	Suit value of cards. 1L is "hearts", 2L is "spades", 3L is "clubs", 4L is "diamonds", and 5L is "trumps". Will be coerced by tarot_suit() .
rank	Rank value of cards. Normal suits are from 1L (ace) to 14L (king) plus a 15L joker. Trump suits are from 1L to 21L plus a 22L fool/excuse. Will be coerced by tarot_rank() .
x, y	Cartesian coordinates (numeric vectors)
angle	Rotation of piece (numeric vector of degrees, counter-clockwise). Will be coerced by piece_angle() .
length.out	The number of pieces. Not needed if all the arguments are the same length (or of length one) and this length is the same as the number of desired pieces.

Value

A [tibble::tibble\(\)](#) data frame with the following columns:

- "piece_side"
- "suit"
- "rank"
- "cfg" (optional, defaults to "piecepack")
- "x"
- "y"
- "angle" (optional, defaults to 0).

This data frame is compatible with [piecepackr::render_piece\(\)](#) and likely [ppcli::cat_piece\(\)](#).

Examples

```
df <- tarot_cards(suit = "clubs", rank = "nine", x = 1.5, y = 2)
if (require("piecepackr", quietly = TRUE) &&
    packageVersion("piecepackr") >= "1.15.0-1" &&
    piecepackr:::device_supports_unicode()) {
  grid::grid.newpage()
  envir = game_systems()
  pmap_piece(df, envir = envir, default.units = "in")
}
```

Index

alquerque_alquerque
 (alquerque_games_variant), 3
alquerque_baghchal
 (alquerque_games_variant), 3
alquerque_bits (alquerque_pieces), 4
alquerque_board (alquerque_pieces), 4
alquerque_games (games_info), 23
alquerque_games_variant, 3
alquerque_none (setup_none), 63
alquerque_pieces, 4
alquerque_setup_by_name
 (setup_by_name), 60
alquerque_suit (ppdf-defunct), 54
as.double(), 33
as.integer(), 34, 35

checker_american_checkers
 (checker_games_variant), 7
checker_american_pool_checkers
 (checker_games_variant), 7
checker_bashni (checker_games_variant),
 7
checker_bits (checker_pieces), 9
checker_board (checker_pieces), 9
checker_brazilian_checkers
 (checker_games_variant), 7
checker_breakthrough
 (checker_games_other), 5
checker_canadian_checkers
 (checker_games_variant), 7
checker_checkers
 (checker_games_variant), 7
checker_column_checkers
 (checker_games_variant), 7
checker_corner_checkers
 (checker_games_variant), 7
checker_crossings
 (checker_games_other), 5
checker_czech_checkers
 (checker_games_variant), 7

checker_dameo (checker_games_variant), 7
checker_dao (checker_games_other), 5
checker_dodgem (checker_games_other), 5
checker_english_checkers
 (checker_games_variant), 7
checker_focus (checker_games_other), 5
checker_four_field_kono
 (checker_games_other), 5
checker_frisian_checkers
 (checker_games_variant), 7
checker_games (games_info), 23
checker_games_other, 5
checker_games_variant, 7
checker_gothic_checkers
 (checker_games_variant), 7
checker_grasshopper
 (checker_games_other), 5
checker_international_checkers
 (checker_games_variant), 7
checker_italian_checkers
 (checker_games_variant), 7
checker_jamaican_checkers
 (checker_games_variant), 7
checker_jeson_mor
 (checker_games_other), 5
checker_julgonu (checker_games_other), 5
checker_kangaroo (checker_games_other),
 5
checker_lasca (checker_games_variant), 7
checker_lines_of_action
 (checker_games_other), 5
checker_malaysian_checkers
 (checker_games_variant), 7
checker_none (setup_none), 63
checker_one_way_checkers
 (checker_games_variant), 7
checker_pieces, 9
checker_portuguese_checkers
 (checker_games_variant), 7

- checker_russian_checkers
 - (checker_games_variant), 7
- checker_setup_by_name (setup_by_name), 60
- checker_singaporean_checkers
 - (checker_games_variant), 7
- checker_spanish_checkers
 - (checker_games_variant), 7
- checker_suit (ppdf-defunct), 54
- checker_thai_checkers
 - (checker_games_variant), 7
- checker_turkish_checkers
 - (checker_games_variant), 7
- checker_unified_pool_checkers
 - (checker_games_variant), 7
- checker_zimbabwean_pool_checkers
 - (checker_games_variant), 7
- checkers_american_checkers
 - (ppdf-defunct), 54
- checkers_american_pool_checkers
 - (ppdf-defunct), 54
- checkers_board (ppdf-defunct), 54
- checkers_brazilian_checkers
 - (ppdf-defunct), 54
- checkers_breakthrough (ppdf-defunct), 54
- checkers_by_name (ppdf-defunct), 54
- checkers_canadian_checkers
 - (ppdf-defunct), 54
- checkers_checkers (ppdf-defunct), 54
- checkers_crossings (ppdf-defunct), 54
- checkers_czech_checkers (ppdf-defunct), 54
- checkers_dameo (ppdf-defunct), 54
- checkers_english_checkers
 - (ppdf-defunct), 54
- checkers_focus (ppdf-defunct), 54
- checkers_four_field_kono
 - (ppdf-defunct), 54
- checkers_frisian_checkers
 - (ppdf-defunct), 54
- checkers_gothic_checkers
 - (ppdf-defunct), 54
- checkers_grasshopper (ppdf-defunct), 54
- checkers_international_checkers
 - (ppdf-defunct), 54
- checkers_italian_checkers
 - (ppdf-defunct), 54
- checkers_jamaican_checkers
 - (ppdf-defunct), 54
- checkers_julgonu (ppdf-defunct), 54
- checkers_lines_of_action
 - (ppdf-defunct), 54
- checkers_none (ppdf-defunct), 54
- checkers_portuguese_checkers
 - (ppdf-defunct), 54
- checkers_russian_checkers
 - (ppdf-defunct), 54
- checkers_spanish_checkers
 - (ppdf-defunct), 54
- checkers_thai_checkers (ppdf-defunct), 54
- checkers_turkish_checkers
 - (ppdf-defunct), 54
- chess_angle (ppdf-defunct), 54
- chess_bits (chess_pieces), 13
- chess_board (chess_pieces), 13
- chess_by_name (ppdf-defunct), 54
- chess_chess (chess_games_variant), 11
- chess_chess960 (chess_games_variant), 11
- chess_fischer_random_chess
 - (chess_games_variant), 11
- chess_games (games_info), 23
- chess_games_variant, 11
- chess_horde_chess
 - (chess_games_variant), 11
- chess_international_chess
 - (chess_games_variant), 11
- chess_jeson_mor (chess_games_variant), 11
- chess_monochrome_chess
 - (chess_games_variant), 11
- chess_none (setup_none), 63
- chess_pieces, 13
- chess_racing_kings
 - (chess_games_variant), 11
- chess_rank (piece_rank), 34
- chess_rank(), 14
- chess_setup_by_name (setup_by_name), 60
- chess_suit (ppdf-defunct), 54
- cube_bits (cube_pieces), 15
- cube_pieces, 15
- d10_dice (dice_pieces), 17
- d12_dice (dice_pieces), 17
- d20_dice (dice_pieces), 17
- d4_dice (dice_pieces), 17
- d8_dice (dice_pieces), 17

- dice_dice (dice_pieces), 17
- dice_pieces, 17
- dice_rank (piece_rank), 34
- dice_rank(), 19
- domino_angle (ppdf-defunct), 54
- domino_bee_donimoes
 - (domino_games_variant), 20
- domino_concentration
 - (domino_games_variant), 20
- domino_finder (domino_games_variant), 20
- domino_freecell (domino_games_variant), 20
- domino_fujisan (domino_games_variant), 20
- domino_games (games_info), 23
- domino_games_variant, 20
- domino_luzon (domino_games_variant), 20
- domino_none (setup_none), 63
- domino_patience (domino_games_variant), 20
- domino_rank (ppdf-defunct), 54
- domino_runners (domino_games_variant), 20
- domino_setup_by_name (setup_by_name), 60
- domino_suit (piece_suit), 35
- domino_suit(), 22
- domino_the_jubilee
 - (domino_games_variant), 20
- domino_the_sultan
 - (domino_games_variant), 20
- domino_tiles, 22
- dominoes_by_name (ppdf-defunct), 54
- dominoes_concentration (ppdf-defunct), 54
- dominoes_domino_finder (ppdf-defunct), 54
- dominoes_domino_runners (ppdf-defunct), 54
- dominoes_fujisan (ppdf-defunct), 54
- dominoes_luzon (ppdf-defunct), 54
- dominoes_none (ppdf-defunct), 54
- dominoes_patience (ppdf-defunct), 54
- dominoes_the_jubilee (ppdf-defunct), 54
- dominoes_tiles (ppdf-defunct), 54
- dynGet(), 62
- fudge_dice (dice_pieces), 17
- fudge_dice_rank (piece_rank), 34
- fudge_dice_rank(), 19
- games_checkers (ppdf-defunct), 54
- games_chess (ppdf-defunct), 54
- games_dominoes (ppdf-defunct), 54
- games_info, 23
- games_piecepack (ppdf-defunct), 54
- games_stackpack (ppdf-defunct), 54
- get(), 62
- go_bits (go_pieces), 25
- go_board (go_pieces), 25
- go_connect6 (go_games_variant), 24
- go_games (games_info), 23
- go_games_variant, 24
- go_go (go_games_variant), 24
- go_gomoku (go_games_variant), 24
- go_none (setup_none), 63
- go_pieces, 25
- go_renju (go_games_variant), 24
- go_setup_by_name (setup_by_name), 60
- go_suit (ppdf-defunct), 54
- marble_bits (marble_pieces), 27
- marble_board (marble_pieces), 27
- marble_games (games_info), 23
- marble_none (setup_none), 63
- marble_pieces, 27
- marble_setup_by_name (setup_by_name), 60
- marble_suit (ppdf-defunct), 54
- meeple_bits (meeple_pieces), 29
- meeple_pieces, 29
- morris_bits (morris_pieces), 31
- morris_board (morris_pieces), 31
- morris_five_mens_morris
 - (morris_games_variant), 30
- morris_games (games_info), 23
- morris_games_variant, 30
- morris_lasker_morris
 - (morris_games_variant), 30
- morris_nine_mens_morris
 - (morris_games_variant), 30
- morris_none (setup_none), 63
- morris_pieces, 31
- morris_setup_by_name (setup_by_name), 60
- morris_seven_mens_morris
 - (morris_games_variant), 30
- morris_six_mens_morris
 - (morris_games_variant), 30
- morris_suit (ppdf-defunct), 54
- morris_ten_mens_morris
 - (morris_games_variant), 30

- morris_three_mens_morris
 (morris_games_variant), 30
- morris_twelve_mens_morris
 (morris_games_variant), 30
- normalize_name, 33
- normalize_name(), 62
- numeral_dice (dice_pieces), 17
- percentile_dice (dice_pieces), 17
- percentile_dice_rank (piece_rank), 34
- percentile_dice_rank(), 19
- piece_angle, 33
- piece_angle(), 4, 14, 16, 19, 22, 26, 28, 29, 32, 37, 53, 66
- piece_rank, 34
- piece_rank(), 22, 36, 53
- piece_suit, 35
- piece_suit(), 4, 10, 14, 16, 19, 26, 28, 29, 32, 36, 52, 60
- piecepack_12345ive
 (piecepack_games_other), 46
- piecepack_accordion
 (piecepack_games_original), 41
- piecepack_achi
 (piecepack_games_traditional), 48
- piecepack_alice_chess
 (piecepack_games_chess), 39
- piecepack_alien_city
 (piecepack_games_original), 41
- piecepack_alquerque
 (piecepack_games_traditional), 48
- piecepack_american_checkers
 (piecepack_games_checkers), 37
- piecepack_american_pool_checkers
 (piecepack_games_checkers), 37
- piecepack_angle (ppdf-defunct), 54
- piecepack_awithlalnannai_mosona
 (piecepack_games_traditional), 48
- piecepack_backgammon
 (piecepack_games_traditional), 48
- piecepack_baghchal
 (piecepack_games_traditional), 48
- piecepack_baroque_chess
 (piecepack_games_chess), 39
- piecepack_bashni
 (piecepack_games_checkers), 37
- piecepack_black_pawn_trucking
 (piecepack_games_original), 41
- piecepack_board, 36
- piecepack_brain_burn
 (piecepack_games_original), 41
- piecepack_brandubh
 (piecepack_games_traditional), 48
- piecepack_brazilian_checkers
 (piecepack_games_checkers), 37
- piecepack_breakthrough
 (piecepack_games_other), 46
- piecepack_burbuja
 (piecepack_games_original), 41
- piecepack_by_name (ppdf-defunct), 54
- piecepack_cardinals_guards
 (piecepack_games_original), 41
- piecepack_cell_management
 (piecepack_games_original), 41
- piecepack_change_change
 (piecepack_games_other), 46
- piecepack_chariots
 (piecepack_games_original), 41
- piecepack_chaturaji
 (piecepack_games_chess), 39
- piecepack_checkers
 (piecepack_games_checkers), 37
- piecepack_chess
 (piecepack_games_chess), 39
- piecepack_chess960
 (piecepack_games_chess), 39
- piecepack_chinese_checkers
 (piecepack_games_original), 41
- piecepack_chinese_chess
 (piecepack_games_chess), 39
- piecepack_climbing_man
 (piecepack_games_original), 41
- piecepack_coin_collectors
 (piecepack_games_original), 41
- piecepack_coins (piecepack_pieces), 50
- piecepack_column_checkers
 (piecepack_games_checkers), 37
- piecepack_corner_checkers
 (piecepack_games_checkers), 37

piecepack_cribbage
 (piecepack_games_traditional),
 48
 piecepack_cribbage_board
 (piecepack_games_traditional),
 48
 piecepack_crocodile_hop
 (piecepack_games_original), 41
 piecepack_crossings
 (piecepack_games_other), 46
 piecepack_czech_checkers
 (piecepack_games_checkers), 37
 piecepack_dao (piecepack_games_other),
 46
 piecepack_dara
 (piecepack_games_traditional),
 48
 piecepack_desfases
 (piecepack_games_original), 41
 piecepack_dice (piecepack_pieces), 50
 piecepack_dodgem
 (piecepack_games_other), 46
 piecepack_dominoids
 (piecepack_games_original), 41
 piecepack_donut_board
 (piecepack_board), 36
 piecepack_easy_slider
 (piecepack_games_original), 41
 piecepack_english_checkers
 (piecepack_games_checkers), 37
 piecepack_evade
 (piecepack_games_other), 46
 piecepack_everest
 (piecepack_games_original), 41
 piecepack_fischer_random_chess
 (piecepack_games_chess), 39
 piecepack_four_blind_mice
 (piecepack_games_original), 41
 piecepack_four_field_kono
 (piecepack_games_traditional),
 48
 piecepack_four_seasons_chess
 (piecepack_games_chess), 39
 piecepack_froggy_bottom
 (piecepack_games_original), 41
 piecepack_fujisan
 (piecepack_games_original), 41
 piecepack_galaxy_express
 (piecepack_games_original), 41
 piecepack_games (games_info), 23
 piecepack_games_checkers, 37
 piecepack_games_chess, 39
 piecepack_games_original, 41
 piecepack_games_other, 46
 piecepack_games_traditional, 48
 piecepack_gothic_checkers
 (piecepack_games_checkers), 37
 piecepack_grasshopper
 (piecepack_games_other), 46
 piecepack_halma
 (piecepack_games_original), 41
 piecepack_ice_floe
 (piecepack_games_original), 41
 piecepack_iceberg
 (piecepack_games_original), 41
 piecepack_international_chess
 (piecepack_games_chess), 39
 piecepack_italian_checkers
 (piecepack_games_checkers), 37
 piecepack_jamaican_checkers
 (piecepack_games_checkers), 37
 piecepack_japan
 (piecepack_games_original), 41
 piecepack_japanese_chess
 (piecepack_games_chess), 39
 piecepack_jeson_mor
 (piecepack_games_chess), 39
 piecepack_julgonu
 (piecepack_games_traditional),
 48
 piecepack_kangaroo
 (piecepack_games_other), 46
 piecepack_kings_valley
 (piecepack_games_other), 46
 piecepack_klondike
 (piecepack_games_original), 41
 piecepack_lab_rats
 (piecepack_games_original), 41
 piecepack_landlocked
 (piecepack_games_original), 41
 piecepack_leapfrog
 (piecepack_games_other), 46
 piecepack_ley_lines
 (piecepack_games_original), 41
 piecepack_lines_of_action
 (piecepack_games_other), 46

- piecepack_ludo
 (piecepack_games_traditional),
 48
- piecepack_lukawan
 (piecepack_games_other), 46
- piecepack_matchsticks
 (piecepack_pieces), 50
- piecepack_mathrix
 (piecepack_games_original), 41
- piecepack_minishogi
 (piecepack_games_chess), 39
- piecepack_nine_mens_morris
 (piecepack_games_traditional),
 48
- piecepack_none (setup_none), 63
- piecepack_one_man_thrag
 (piecepack_games_original), 41
- piecepack_one_way_checkers
 (piecepack_games_checkers), 37
- piecepack_pass_the_food
 (piecepack_games_original), 41
- piecepack_pawns (piecepack_pieces), 50
- piecepack_pawns_crossing
 (piecepack_games_original), 41
- piecepack_piece_gaps
 (piecepack_games_original), 41
- piecepack_piece_packing_pirates
 (piecepack_games_original), 41
- piecepack_piecepack_accordion
 (ppdf-defunct), 54
- piecepack_piecepack_halma
 (ppdf-defunct), 54
- piecepack_piecepack_klondike
 (ppdf-defunct), 54
- piecepack_piecepackman
 (piecepack_games_original), 41
- piecepack_pieces, 50
- piecepack_plans_of_action
 (piecepack_games_original), 41
- piecepack_portuguese_checkers
 (piecepack_games_checkers), 37
- piecepack_preview, 53
- piecepack_pyramids (piecepack_pieces),
 50
- piecepack_quatri
 (piecepack_games_other), 46
- piecepack_racing_kings
 (piecepack_games_chess), 39
- piecepack_rank (ppdf-defunct), 54
- piecepack_rect_board_tiles
 (ppdf-defunct), 54
- piecepack_rectangular_board
 (piecepack_board), 36
- piecepack_relativity
 (piecepack_games_original), 41
- piecepack_russian_checkers
 (piecepack_games_checkers), 37
- piecepack_salta
 (piecepack_games_traditional),
 48
- piecepack_san_andreas
 (piecepack_games_original), 41
- piecepack_sarcophagus
 (piecepack_games_original), 41
- piecepack_saucers (piecepack_pieces), 50
- piecepack_seega
 (piecepack_games_traditional),
 48
- piecepack_setup_by_name
 (setup_by_name), 60
- piecepack_ship_it
 (piecepack_games_original), 41
- piecepack_shogi
 (piecepack_games_chess), 39
- piecepack_shopping_mall
 (piecepack_games_original), 41
- piecepack_skyscrapers
 (piecepack_games_original), 41
- piecepack_slides_of_action
 (piecepack_games_original), 41
- piecepack_spanish_checkers
 (piecepack_games_checkers), 37
- piecepack_speedy_towers
 (piecepack_games_original), 41
- piecepack_steppin_stones
 (piecepack_games_original), 41
- piecepack_suit (ppdf-defunct), 54
- piecepack_tablan
 (piecepack_games_traditional),
 48
- piecepack_tablut
 (piecepack_games_traditional),
 48
- piecepack_tariffs
 (piecepack_games_original), 41
- piecepack_thai_checkers

- (piecepack_games_checkers), 37
- piecepack_the_in_crowd
 - (piecepack_games_original), 41
- piecepack_the_magic_bag
 - (piecepack_games_original), 41
- piecepack_the_penguin_game
 - (piecepack_games_original), 41
- piecepack_three_musketeers
 - (piecepack_games_other), 46
- piecepack_tic_tac_toe
 - (piecepack_games_traditional), 48
- piecepack_tiles (piecepack_pieces), 50
- piecepack_tornado_vs_trailer_park
 - (piecepack_games_original), 41
- piecepack_tower_of_babel
 - (piecepack_games_original), 41
- piecepack_tracers
 - (piecepack_games_original), 41
- piecepack_triactor
 - (piecepack_games_original), 41
- piecepack_tula
 - (piecepack_games_original), 41
- piecepack_turkish_checkers
 - (piecepack_games_checkers), 37
- piecepack_twelve_mens_morris
 - (piecepack_games_traditional), 48
- piecepack_ultima
 - (piecepack_games_chess), 39
- piecepack_unified_pool_checkers
 - (piecepack_games_checkers), 37
- piecepack_wormholes
 - (piecepack_games_original), 41
- piecepack_xiangqi
 - (piecepack_games_chess), 39
- piecepack_yote
 - (piecepack_games_traditional), 48
- piecepack_zimbabwean_pool_checkers
 - (piecepack_games_checkers), 37
- piecepackr::game_systems(), 10, 14, 33–35
- piecepackr::render_piece(), 3, 4, 6, 9, 11, 13, 15, 16, 19, 21, 23, 25, 26, 28, 29, 31, 32, 37, 39, 41, 46, 48, 50, 53, 54, 58, 60, 62, 65, 66
- ppcli::cat_piece(), 3, 4, 6, 9, 11, 13, 15, 16, 19, 21, 23, 25, 26, 28, 29, 31, 32, 37, 39, 41, 46, 48, 50, 53, 54, 58, 60, 62, 65, 66
- ppdf-defunct, 54
- ppn::read_ppn(), 33–35
- reversi_bits (reversi_pieces), 59
- reversi_board (reversi_pieces), 59
- reversi_games (games_info), 23
- reversi_games_variant, 58
- reversi_ming_mang
 - (reversi_games_variant), 58
- reversi_none (setup_none), 63
- reversi_pieces, 59
- reversi_reversi
 - (reversi_games_variant), 58
- reversi_setup_by_name (setup_by_name), 60
- setup_by_name, 60
- setup_none, 63
- stackpack_alice_chess
 - (stackpack_games_other), 64
- stackpack_baroque_chess
 - (stackpack_games_other), 64
- stackpack_by_name (ppdf-defunct), 54
- stackpack_chaturaji
 - (stackpack_games_other), 64
- stackpack_chess
 - (stackpack_games_other), 64
- stackpack_chess960
 - (stackpack_games_other), 64
- stackpack_chinese_chess
 - (stackpack_games_other), 64
- stackpack_fischer_random_chess
 - (stackpack_games_other), 64
- stackpack_four_seasons_chess
 - (stackpack_games_other), 64
- stackpack_games (games_info), 23
- stackpack_games_other, 64
- stackpack_horde_chess
 - (stackpack_games_other), 64
- stackpack_international_chess
 - (stackpack_games_other), 64
- stackpack_japanese_chess
 - (stackpack_games_other), 64
- stackpack_none (setup_none), 63
- stackpack_salta
 - (stackpack_games_other), 64

stackpack_setup_by_name
 (setup_by_name), 60
stackpack_shogi
 (stackpack_games_other), 64
stackpack_ultima
 (stackpack_games_other), 64
stackpack_xiangqi
 (stackpack_games_other), 64

tarot_angle (ppdf-defunct), 54
tarot_cards (tarot_pieces), 65
tarot_games (games_info), 23
tarot_none (setup_none), 63
tarot_pieces, 65
tarot_rank (piece_rank), 34
tarot_rank(), 66
tarot_setup_by_name (setup_by_name), 60
tarot_suit (piece_suit), 35
tarot_suit(), 66
tibble::tibble(), 3, 4, 6, 9, 11, 13, 15, 16,
 19, 21, 22, 24–26, 28, 29, 31, 32, 37,
 39, 41, 46, 47, 50, 53, 54, 58, 60, 62,
 65, 66