Package 'bridger'

October 12, 2022

Type Package

Title Bridge Hand Generator with Criteria Selector

Version 0.1.0

Author Jason Kaplan [aut, cre]

URL https://github.com/CommoditiesAI/bridger

Maintainer Jason Kaplan <scjase@gmail.com>

Description
Produce bridge hands, allowing parameters for hands to offer specific for bidding sequences.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.1.1

Imports cowplot, dplyr, patchwork, tibble, tidyr, magrittr, ggplot2, ggedit, glue, gridExtra, kableExtra, pdftools, scales, stringr

SystemRequirements LaTeX(texi2dvi) must be present in the system to create PDF reports

Depends R (>= 2.10)

Suggests spelling

Language en-US

NeedsCompilation no

Repository CRAN

Date/Publication 2021-08-24 20:10:02 UTC

R topics documented:

`.onLoad` *zzz*

Description

Runs on loading bridger

Usage

```
.onLoad(libname, pkgname)
```

Arguments

`libname`	Legacy dummy
`pkgname`	Legacy dummy

Value

No return value, called to set global variables and specify import packages

bridgeHand *bridgeHand*

Description

Generate a bridge hand

Usage

```
bridgeHand(
  handNumber = "auto",
  seat = FALSE,
  createGraphic = TRUE,
  LTC = "original",
  ...
)
```

Arguments

handNumber	An integer for generating a hand, or "auto" to use a random number generator
seat	If not false, makes the specified seat South and dealer, so all bidding starts with South and the specified hand type
createGraphic	Whether the graphic should be created
LTC	Whether to include losing trick count - FALSE for none, "original" or "new" for schema
...	Other parameters used in hand evaluation

Value

List: Hand ID, Dealer, Hand graphic, Hand points, Hand shape, vulnerability

Note

To change the hand evaluation pass high card values (HCValues) and shape values (shapeValues) in the arguments.

HCValues is a string of five digits specifying the value of the Ace, King, Queen, Jack and 10. The default is the Milton Work scale of 4, 3, 2, 1, 0. shapeValues is a string of eight digits specifying the value of a suit with no cards/"Void", 1-card/"Singleton", ... 7-cards. The default is c(3, 2, 1, 0, 0, 1, 2, 3) Losing Trick Count (LTCSchema) 'Original' or 'New' as described at https://en.wikipedia.org/wiki/Losing-Trick_Count. This assumes a fit will be found. It is currently not implemented.

Examples

```
## Not run:
# Produce a bridge hand
hand <- bridgeHand()

# Produce a bridge hand '500' ensuring South as dealer
hand500 <- bridgeHand(handNumber = 500, seat = "S") # Seat can be any compass point

## End(Not run)
```

`collectHands` *collectHands*

Description

Returns a list of hands that fit a requirement. Simple hands will most often give the required bids. Complex hands, where a subsequent bid is made, may not fit the requirements, as other bids by opponents or partner may be preferable to the desired bidding pattern.

Usage

```
collectHands(handType = "opener", num = 6, ...)
```

Arguments

`handType`	Type of hands wanted
`num`	Number of hands requested
`...`	Other parameters to be passed to the find_ functions, e.g. HC_low, cardLen_low

Value

Tibble - One line per requested hand with hand ID, seat position and type of hand

Note

Each of the handTypes is a standard set of parameters. For example "NT" (alias "balanced") allows 12-14 points, a single doubleton and no 5-card majors and no 6-card minor. To change these parameters then optional parameters can be passed through the "...". The most common changes will be to specify the low and high high-card range and the shortest allowed suit and longest allowed. These are "HC_low" and "HC_high", "cardLen_low" and "cardLen_high" respectively.

Existing functions and key parameters are currently:

Single bids

	HC_low	HC_high	cardLen_low	cardLen_high
any	0	40	0	13
opener	12	40	0	13
1major	12	19	4 (Major) Any (Minor)	13

1NT	12	14	2	4
4441	12	40	1	4
strong	19	40	0	8
preempt2	5	10	0	6
preempt3	6	9	0	7
			0	7
Complex bids	6	9	0	7
	South	West	North	East
1NT_LHOdouble	1NT	X		
1NT_LHObid	1NT	Any		
1NT_RHObid	1NT	Pass	Pass	Any
1major_jacoby2NT	1major	Pass	2NT(Jacoby)	

Other parameters are also used, but individually assigned in the function.

Examples

```
## Not run:
# Collect the ids of 2 hands with any shape
hands <- collectHands(num = 2)
# Collect 6 hands with opening points and a "4441" shape
hands <- collectHands(handType = "4441", num = 6)

# Collect a weak no-trump hand, with a point range of 11 to 15
hands <- collectHands(handType = "weakNT", num = 1, HC_low = 11, HC_high = 15)

## End(Not run)
```

`createGraphic` *createGraphic*

Description

Create the graphic of the hand

Usage

```
createGraphic(handNo, handN, handE, handS, handW, dealer, vuln, points)
```

Arguments

handNo	The id of the hand
handN	The North hand generated by bridgeHand
handE	The East hand generated by bridgeHand
handS	The South hand generated by bridgeHand

handW	The West hand generated by bridgeHand
dealer	The hand to become South, the designated dealer
vuln	The hand's vulnerability
points	The hand's points

Value

ggplot graphic object

`find_1major` *find_1major*

Description

Return a bridge hand that will open 1 of a major

Assumes that a 5 card minor will be bid before 4 card major, except if "canape" set to TRUE, then a 6 card minor will be opened before a 4 card major

Assumes a weak 1NT, so HC_low is the first point outside the range of 1NT.

Usage

```
find_1major(HC_low = 15, HC_high = 19, cardLen_min = 4, canape = FALSE)
```

Arguments

HC_low	The minimum number of high-card points
HC_high	The maximum number of high-card points, otherwise 2-level bid is possible
cardLen_min	The minimum number of cards in the major
canape	Whether a 4 card major will be opened before a 5 card minor

Value

id and seat of compliant hand

find_1major_jacoby2NT *find_1major_jacoby2NT*

Description

Find hands where South opens one of a major, and North will bid 2NT, to show 4 card support and points for game

Usage

```
find_1major_jacoby2NT(HC_low = 13, cardLen_low = 4)
```

Arguments

HC_low	The minimum number of high-card points
cardLen_low	The minimum length of a suit

Value

id and seat of a compliant hand

find_2preempt *find_2preempt*

Description

Find hands that are likely to preempt at the 2 level in a major

Usage

```
find_2preempt(HC_low = 5, HC_high = 10, cardLen_low = 6, cardLen_high = 7)
```

Arguments

HC_low	The minimum number of high-card points
HC_high	The maximum number of high-card points
cardLen_low	The minimum length of a suit
cardLen_high	The maximum length of a suit

Value

id and seat of compliant hand

`find_3preempt` *find_3preempt*

Description

Find hands that are likely to preempt at the 3 level

Usage

```
find_3preempt(HC_low = 5, HC_high = 10, cardLen_low = 7, cardLen_high = 8)
```

Arguments

`HC_low`	The minimum number of high-card points
`HC_high`	The maximum number of high-card points
`cardLen_low`	The minimum length of a suit
`cardLen_high`	The maximum length of a suit

Value

FALSE if not compliant, or id and seat of compliant hand

`find_4441` *find_4441*

Description

Find hands that comply with a 4441 shape and opening point count

Usage

```
find_4441(HC_low = 12, HC_high = 35, cardLen_low = 5, cardLen_high = 13)
```

Arguments

`HC_low`	The minimum number of high-card points
`HC_high`	The maximum number of high-card points
`cardLen_low`	The minimum length of a suit
`cardLen_high`	The maximum length of a suit

Value

id and seat of compliant hand

find_any *find_any*

Description

Return any bridge hand - May not be an opener

Usage

```
find_any()
```

Value

id and seat of compliant hand

find_opener *find_opener*

Description

Return a bridge hand that is likely to open

Usage

```
find_opener(HC_low = 12)
```

Arguments

HC_low	The minimum number of high-card points

Value

id and seat of compliant hand

find_strong *find_strong*

Description

Find hands that are strong enough to open strong

Usage

```
find_strong(HC_low = 19, HC_high = 35, cardLen_low = 1, cardLen_high = 5)
```

Arguments

HC_low	The minimum number of high-card points
HC_high	The maximum number of high-card points
cardLen_low	The minimum length of a suit
cardLen_high	The maximum length of a suit

Value

id and seat of compliant hand

find_strongNT *find_strongNT*

Description

Find hands that comply with a weak no trump opening

Usage

```
find_strongNT(HC_low = 15, HC_high = 17, cardLen_low = 2, cardLen_high = 5)
```

Arguments

HC_low	The minimum number of high-card points
HC_high	The maximum number of high-card points
cardLen_low	The minimum length of a suit
cardLen_high	The maximum length of a suit

Value

id and seat of compliant hand

find_weak1NT_LHObid *find_weak1NT_LHObid*

Description

Find hands where South will open a weak 1NT and West will likely bid

Usage

```
find_weak1NT_LHObid(HC_low = 7, cardLen_low = 6)
```

Arguments

HC_low	The minimum number of high-card points
cardLen_low	The minimum length of a suit

Value

id and seat of a compliant hand

find_weak1NT_LHOx *find_weak1NT_LHOx*

Description

Find hands where South will open a weak 1NT and West will likely double

Usage

```
find_weak1NT_LHOx(
  HC_low = 12,
  HC_high = 14,
  cardLen_low = 2,
  cardLen_high = 5,
  pointsForDouble = 15
)
```

Arguments

HC_low	The minimum number of high-card points
HC_high	The maximum number of high-card points
cardLen_low	The minimum length of a suit
cardLen_high	The maximum length of a suit
pointsForDouble	Minimum number of points for West to double

Value

id and seat of a compliant hand

`find_weak1NT_RHObid` *find_weak1NT_RHObid*

Description

Find hands where South will open a weak 1NT, East and North with pass, and West will likely bid

Usage

```
find_weak1NT_RHObid(HC_low = 7, cardLen_low = 6)
```

Arguments

`HC_low`	The minimum number of high-card points
`cardLen_low`	The minimum length of a suit

Value

id and seat of a compliant hand

id and seat of a compliant hand

`find_weakNT` *find_weakNT*

Description

Find hands that comply with a no trump opening

Usage

```
find_weakNT(HC_low = 12, HC_high = 14, cardLen_low = 2, cardLen_high = 4)
```

Arguments

`HC_low`	The minimum number of high-card points
`HC_high`	The maximum number of high-card points
`cardLen_low`	The minimum length of a suit
`cardLen_high`	The maximum length of a suit

Value

id and seat of compliant hand

printHands *printHands*

Description

Produce a page of bridge hands as a PDF. Each page can hold up to 6 hands, and can show all seats or one of the seats can be selected through the 'outputSeats' parameter.

- "FULL" or "F" - Show all seats.
- "N" / "E" / "S" / "W" - Show only the specified seats on separate outputs. e.g. "NS" to generate North and South seats.
- "ALL" or "A" - Equivalent to "FNEWS", i.e. Separate pages of each of the four seats, and one page with all seats.

In all cases, only point counts for the selected seats will be visible.

The output PDFs will be saved to a temporary directory, but a directory can be specified in the 'saveOutput' parameter.

Usage

```
printHands(
  ids = FALSE,
  seats = FALSE,
  handType = "any",
  num = 12,
  outputSeats = "F",
  saveOutputDir = FALSE,
  ...
)
```

Arguments

ids	The ids of hands to be generated
seats	The seats of the hands in ids, i.e. the seat which gives the requested conditions, this will become South when printed
handType	The type of hand required, default is 'any'. Alternatives include, '4441', 'strong', ...
num	The number of hands wanted
outputSeats	Character code of required seats, "N", "E", "S", "W" and "F" for the full hand NB "ALL" equivalent to "FNEWS"
saveOutputDir	If FALSE (Default) will save to temporary directory, or specify a directory, e.g. "c:/temp/bridger"
...	Other variables that may be passed when selecting compliant hands

Value

Text message, confirming completion and specifying location of PDF outputs

Examples

```
## Not run:
# Produce a hand showing all seats and save them to 'c:/temp/bridger' directory
printHands(handType = "any", num = 1, outputSeats = "FULL", saveOutput = FALSE)
# Produce a page of 6 hands likely to open with a 3-level preempt, only showing the South seat
printHands(handType = "preempt3", num = 6, outputSeats = "S")

# Produce the specified hands, showing all seats
printHands(ids = c(500, 501, 502), seats = c("E", "W", "S"), outputSeats = "FULL")

## End(Not run)
```

---

suitSplit *suitSplit*

Description

Provides the probabilities with with a number of cards will split between two hands, given a number of unknown cards in each hand. Unknown hands are assumed to be West and East.

If there is no information to indicate different numbers of unknown cards in both hands, then symmetrical probabilities will be returned. However, if one hand is expected to have a different number of cards to the other, then these can be specified. For example, if during the bidding East overcalled in spades, indicating a 5 card suit, then when looking at hearts, East has fewer cards. While the number of assumed cards in West's hand is 13 ('cards_W = 13'), the assumed cards in East should be reduced to 8 ('cards_E = 8')

Usage

```
suitSplit(missingCards = 5, cards_W = 13, cards_E = 13)
```

Arguments

missingCards	The number of cards held by the two hands
cards_W	Cards in West hands
cards_E	Cards in East hands

Value

Tibble of probabilities

Examples

```
suitSplit(missingCards = 6, cards_W = 13, cards_E = 8)
```

Index