Package 'poker'

August 9, 2017

Title Play Texas Hold Em Poker

Version 0.8.8

Date 2017-08-8

Author Benjamin Greenspan [aut,cre]

Maintainer Benjamin Greenspan <Benjamin.Greenspan@gmail.com>

Description Type testRoundOfPoker() to demonstrate the game of Texas Hold 'Em poker. Rotate the dealer button, deal cards, rank each hand, compare ranks, break ties (if necessary), determine the winner, output a textual summary, and output a graphical user interface.

Depends

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-08-09 04:08:33 UTC

R topics documented:

`assignToBoard` *assignToBoard*

Description

Deal 3 community cards.

Usage

```
assignToBoard(y)
```

Arguments

y	cards dealt as vector[2*nPlayers+3] in {1, 2, ..., 52}

Value

board : the board cards as vector[5] in {1, 2, ..., 52}

Examples

```
assignToBoard(1:23)
assignToBoard(c(1:17,24,48:52))
```

`assignToPlayers` *assignToPlayers*

Description

A standard deal situation beginnng the deal at the left of the dealer.

Usage

```
assignToPlayers(nPlayers, position, y)
```

Arguments

`nPlayers`	number of hands to deal as integer in {2, ... , 9}
`position`	dealer position as integer in {2, ..., nPlayers}
`y`	cards dealt as vector[2*nPlayers+5] in {1, 2, ..., 52}

Value

players : the hole cards in absolute position	as matrix[nPlayers, 4] in {1, 2, ..., 52} col1: rank of card 1 in {2, ... , 14} col2: suit of card 1 in {1, 2, 3, 4} col3: rank of card 2 col4: suit of card 2

See Also

`dotTransformToAbsolute`

Examples

```
assignToPlayers(9,9,1:23)
assignToPlayers(9,1,1:23)
assignToPlayers(9,1,c(1:17,24,48:52))
```

`cgiPlayers` *cgiPlayers*

Description

A primitive method (i.e., does not support classes) for graphics using the plot() function. Built-in support for 2-9 players. This function was written on a Mac and may not be PC-compatible (yet). You must have already called cgiPlayers(time=1, ...) before calling cgiPlayers(time=2, ...), you must have already called cgiPlayers(time=1, ...) and cgiPlayers(time=2, ...) before calling cgiPlayers(time=3, ...), etc.

Usage

```
cgiPlayers(time, alias, position, cards)
```

Arguments

`time` :

the current round as integer in {1, 2, 3, 4}

1 = pre-flop
2 = flop
3 = turn
4 = river

`alias` names of players as vector[nPlayers]

`position` dealer position as integer in {2, ..., nPlayers}

`cards` :

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

Value

In lieu of a return value, cgiPlayers calls the plot() function.

Examples

```
alias <- c("Player1","Player2","Player3","Player4","Player5")
alias <- c(alias,"Player6","Player7","Player8","Player9")
cols1thru5 <- c(2,8,12,14,10,6,14,8,4,2,3,2,4,1,4,3,1,1,13,4,4,5,3,9,8,12,7)
cols1thru5 <- c(cols1thru5,3,4,3,2,2,4,2,1,1,3,3,3,3,3,3,3,3)
cols6thru10 <- c(1,1,1,1,1,1,1,1,1,10,10,10,10,10,10,10,10,10,4,4,4,4,4,4,4,4,4)
cols6thru10 <- c(cols6thru10,12,12,12,12,12,12,12,12,12,4,4,4,4,4,4,4,4,4)
cols11thru14 <- c(11,11,11,11,11,11,11,11,11,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2)
```

```
cols11thru14 <- c(cols11thru14,3,3,3,3,3,3,3,3,3)
cards <- matrix(c(cols1thru5,cols6thru10,cols11thru14),nrow=9,ncol=14); cards
cgiPlayers(1,alias,9,cards)
cgiPlayers(2,alias,9,cards)
cgiPlayers(3,alias,9,cards)
cgiPlayers(4,alias,9,cards)
```

`deal` *deal*

Description

Generate Player+Community cards = 2x(nPlayers)+5 cards.

Usage

```
deal(nPlayers, position)
```

Arguments

`nPlayers`	number of hands to deal as integer in {2, ... , 9}
`position`	dealer position as integer in {2, ..., nPlayers}

Value

y : cards dealt in hole as vector[nCards] in {1, 2, ..., 52}

Examples

```
deal(9,9)
deal(9,1)
```

`dotFlush` *dotFlush*

Description

Determine the player with the highest flush.

Usage

```
dotFlush(cards, score)
```

Arguments

`cards`	:

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

`score` :

the score of the hand in absolute terms as vector[nPlayers]

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

Value

winner : absolute position of the winner as vector

See Also

`dotFlushRanker` and `dotHighcardCompare`

Examples

```
cards <- c(2,1,3,3,5,2,6,3,7,3,13,3,14,3,2,3,3,4,5,1,6,3,7,3,13,3,14,3)
cards <- matrix(cards,2,14,byrow=TRUE); cards
score <- showdown(cards); score
dotFlush(cards,score)

cards <- c(2,1,3,3,5,3,6,3,7,3,13,3,14,3,2,3,3,4,5,3,6,3,7,3,13,3,14,3)
cards <- matrix(cards,2,14,byrow=TRUE);cards
score <- showdown(cards); score
dotFlush(cards,score)
```

dotFlushRanker *dotFlushRanker*

Description

Return the ranks of the 5 highest cards in the flush.

Usage

```
dotFlushRanker(cardsRow)
```

Arguments

cardsRow :

one 7 card hand as vector[14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

Value

flushRank : the rank of 5 high cards in flush as vector[5]

col1: suit of card 1 in {2, ... , 14}
.
.
.
col5: suit of card n in {2, ... , 14}

Examples

```
dotFlushRanker(c(2,1,2,2,5,2,7,2,8,2,9,2,11,1))
dotFlushRanker(c(2,1,2,2,5,2,7,2,8,2,9,2,11,2))
```

dotFourOfAKind *dotFourOfAKind*

Description

Determine the player with the highest hand (i.e., four of a kind and kicker) with score of 8.

Usage

```
dotFourOfAKind(nPlayers, cards, score)
```

Arguments

`nPlayers` : number of hands to deal as integer in {2, ... , 9}

`cards` :

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

`score` :

the score of the hand in absolute terms as vector[nPlayers]

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

Value

winner : absolute position of the winner as vector

See Also

`dotFourOfAKindRanker`

Examples

```
cards <- c(14,10,5,1,2,1,14,9,7,2,2,2,4,4,4,3,3,3,8,8,8,3,3,3,13,13,13)
cards <- c(cards,3,3,3,14,14,14,3,3,3,14,14,14,4,4,4)
cards <- matrix(cards,nrow=3,ncol=14); cards
```

```
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
dotFourOfAKind(nPlayers,cards,score)

cards <- c(3,4,5,1,1,1,8,9,10,1,1,1,14,14,14,1,1,1,14,14,14,2,2,2,7,7,7)
cards <- c(cards,3,3,3,14,14,14,3,3,3,14,14,14,4,4,4)
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
dotFourOfAKind(nPlayers,cards,score)

cards <- c(3,4,5,1,1,1,8,9,10,1,1,1,14,14,14,1,1,1,14,14,14,2,2,2,11,11,11)
cards <- c(cards,3,3,3,14,14,14,3,3,3,14,14,14,4,4,4)
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
dotFourOfAKind(nPlayers,cards,score)
```

`dotFourOfAKindRanker` *dotFourOfAKindRanker*

Description

Determine the rank of the four of a kind and the kicker. This functions assumes ranks are sorted in decreasing order.

Usage

```
dotFourOfAKindRanker(oneHand)
```

Arguments

`oneHand` :

the ranks of one 7 card hand as vector[7]

col1: rank of card 1 in {2, ... , 14}
col2: rank of card 2
.
.
.
col7: rank of card 7

Value

fourOfAKindRank : the ranks of the quads and the high kicker as vector

col1: the rank of the quads
col2: the rank of the kicker

Examples

```
dotFourOfAKindRanker(c(14,14,14,14,10,7,6))
dotFourOfAKindRanker(sort(c(10,14,6,14,7,14,14),decreasing=TRUE))
```

`dotFullHouse` *dotFullHouse*

Description

Determine the player with the highest boat.

Usage

```
dotFullHouse(cards, score)
```

Arguments

`cards` :

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

`score` :

the score of the hand in absolute terms as vector[nPlayers]

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

Value

winner : absolute position of the winner as vector

See Also

dotFullHouseRanker

Examples

```
cards <- c(5,10,4,8,1,2,1,1,10,9,6,7,3,2,2,2,5,5,5,5,3,3,3,3,8,8,8,8,3,3,3,3)
cards <- c(cards,14,14,14,14,2,2,2,2,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
dotFullHouse(cards,score)

cards <- c(5,10,4,8,1,2,1,1,10,9,6,7,3,2,2,2,12,12,12,12,1,1,1,1,12,12,12,12,3,3,3,3)
cards <- c(cards,14,14,14,14,2,2,2,2,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
dotFullHouse(cards,score)
```

---

dotFullHouseRanker *dotFullHouseRanker*

Description

Determine the rank of the top set and the top pair.

Usage

```
dotFullHouseRanker(oneHand)
```

Arguments

oneHand :

the ranks of one 7 card hand as vector[7]

col1: rank of card 1 in {2, ... , 14}
col2: rank of card 2
.
.
.
col7: rank of card 7

Value

fullHouseRank : the ranks of the high set and the high pair as vector

col1: the rank of the top set

col2: the rank of the top pair

Examples

```
dotFullHouseRanker(c(2,2,2,5,5,8,9))
dotFullHouseRanker(c(2,2,5,5,5,8,9))
dotFullHouseRanker(c(2,2,5,5,5,8,8))
```

---

dotHighcard *dotHighcard*

Description

Determine the player(s) with a high card hand.

Usage

```
dotHighcard(cards)
```

Arguments

cards :

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

Value

winner : absolute position of the winner as vector

See Also

dotHighcardCompare

Examples

```
dotHighcard(matrix(c(2,1,14,2,5,3,6,4,7,1,13,2,14,3,2,3,3,4,5,1,6,2,7,3,13,4,14,1),2,14,byrow=TRUE))
dotHighcard(matrix(c(2,1,3,2,5,3,6,4,7,1,13,2,14,3,2,3,3,4,5,1,6,2,7,3,13,4,14,1),2,14,byrow=TRUE))
```

dotHighcardCompare *dotHighcardCompare*

Description

Determine the player(s) with the high card.

Usage

```
dotHighcardCompare(rankMatrix)
```

Arguments

rankMatrix	:

the ranks from the 7 card hand as matrix[nPlayers, 7]

col1: rank of card 1 in {2, ... , 14}
col2: rank of card 2
.
.
.
col7: rank of card 7

Value

winner : absolute position of the winner as vector

Examples

```
dotHighcardCompare(matrix(c(2,4,5,6,7,13,14,2,3,5,6,7,13,14),2,7,byrow=TRUE))
dotHighcardCompare(matrix(c(2,3,5,6,7,13,14,2,3,5,6,7,13,14),2,7,byrow=TRUE))
```

dotPair *dotPair*

Description

Determine the player(s) with the highest pair and kicker cards.

Usage

```
dotPair(nPlayers, cards, score)
```

Arguments

nPlayers	number of hands as integer in {2, ... , 9}
cards	:

the 7 card hands as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

`score` :

the score of the hands in absolute terms as vector[nPlayers]

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

Value

winner : absolute position of the winner as vector

See Also

`dotPairRanker` and `dotHighcardCompare`

Examples

```
cards <- c(2,3,4,1,1,1,2,3,6,2,2,2,4,4,4,3,3,3,11,11,11,3,3,3,13,13,13)
cards <- c(cards,3,3,3,14,14,14,3,3,3,9,9,9,4,4,4)
cards <- matrix(cards,nrow=3,ncol=14)
dotPair(3,cards,c(2,2,2))
```

`dotPairRanker` *dotPairRanker*

Description

Determine the rank of the pair. Notes: dotPairRanker requires a hand with a score of 2 (i.e., a pair). This functions works best when ranks are sorted in decreasing order.

Usage

```
dotPairRanker(oneHand)
```

Arguments

oneHand :

a sorted hand with ranks only as vector[7]

col1: rank of card 1 in {2, ... , 14}
col2: rank of card 2
.
.
.
col7: rank of card 7

Value

pairRank : the rank of the pair as vector

Examples

```
dotPairRanker(c(2,2,5,6,7,13,14))
```

---

dotScorer *dotScorer*

Description

Determine the ranking of one hand.

Usage

```
dotScorer(cardsRow)
```

Arguments

cardsRow :

one 7 card hand as vector[14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

Value

ranking : the rank of the hand as integer in {2, ... , 9}

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

See Also

dotTransformToNumber, dotTransformToRank

Examples

```
dotScorer(c(2,1,3,2,5,3,6,4,7,1,13,2,14,2))
dotScorer(c(2,1,2,2,5,3,6,4,7,1,13,2,14,2))
dotScorer(c(2,1,2,2,5,3,5,4,7,1,13,2,14,2))
dotScorer(c(2,1,2,2,2,3,5,4,7,1,13,2,14,2))
dotScorer(c(2,1,3,2,4,3,5,4,6,1,13,2,14,2))
dotScorer(c(2,1,3,1,5,1,6,1,7,1,13,2,14,2))
dotScorer(c(2,1,2,2,2,8,13,8,7,1,13,2,14,2))
dotScorer(c(2,1,2,2,2,3,2,4,7,1,13,2,14,2))
dotScorer(c(2,1,3,1,4,1,5,1,6,1,7,1,14,2))
```

`dotStraight` *dotStraight*

Description

Determine the player with the highest straight.

Usage

```
dotStraight(cards, score)
```

Arguments

`cards` :

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

`score` :

the score of the hand in absolute terms as vector[nPlayers]

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

Value

winner : absolute position of the winner as vector

See Also

dotStraightRanker

Examples

```
cards <- c(7,1,4,2,4,1,4,3,10,1,11,2,2,2,2,3,3,3,3,3,3,1,1,1,5,5,5)
cards <- c(cards,4,4,4,6,6,6,2,2,2,14,14,14,2,2,2)
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
dotStraight(cards, score)

cards <- c(2,1,4,2,4,1,4,3,10,1,11,2,2,2,2,3,3,3,3,3,3,1,1,1,5,5,5)
cards <- c(cards,4,4,4,6,6,6,2,2,2,14,14,14,2,2,2)
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
dotStraight(cards, score)
```

`dotStraightFlush` *dotStraightFlush*

Description

Determine the player with the highest straight flush.

Usage

```
dotStraightFlush(nPlayers, cards, score)
```

Arguments

`nPlayers` number of hands as integer in {2, ... , 9}

`cards` :

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

`score` :

the score of the hand in absolute terms as vector[nPlayers]

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

Value

winner : absolute position of the winner as vector

See Also

`dotTransformToNumber` and `dotStraightFlushRanker`

Examples

```
cards <- c(8,13,5,1,1,4,6,2,2,2,3,4,14,14,14,2,2,2,9,9,9,1,1,1,10,10,10)
cards <- c(cards,1,1,1,11,11,11,1,1,1,12,12,12,1,1,1)
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
dotStraightFlush(nPlayers,cards,score)

cards <- c(1,1,3,4,2,2,3,4,8,8,1,1,9,9,1,1,10,10,1,1,11,11,1,1,12,12,1,1)
cards <- matrix(cards,nrow=2,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
dotStraightFlush(nPlayers,cards,score)
```

`dotStraightFlushRanker` *dotStraightFlushRanker*

Description

Determine the rank of the highest card in a straight flush. This function assumes cards are sorted in ascending order.

Usage

```
dotStraightFlushRanker(yTempRow)
```

Arguments

`yTempRow` :

a sorted 7 card hand of numbers as vector[7]

col1: number of card 1 in {1, 2, ... , 52}
col2: number of card 2
.
.
.
col7: number of card 7

Value

straightFlushRank : the top card in the straight flush as integer

Examples

```
dotStraightFlushRanker(c(1,2,3,4,5,15,19))
dotStraightFlushRanker(c(9,10,11,12,13,35,42))
dotStraightFlushRanker(c(9,10,11,12,13,14,35))
```

```
dotStraightFlushRanker(c(1,2,3,4,13,20,35))
dotStraightFlushRanker(c(9,26,14,15,16,17,35))
```

`dotStraightRanker` *dotStraightRanker*

Description

Returns the rank of the highest card in the straight.

Usage

```
dotStraightRanker(oneHand)
```

Arguments

`oneHand` :

a sorted hand with ranks only as vector[7]

col1: rank of card 1 in {2, ... , 14}
col2: rank of card 2
.
.
.
col7: rank of card 7

Value

straightRank : the rank of top card in the straight as integer

Examples

```
dotStraightRanker(c(2,3,4,5,6,9,10))
dotStraightRanker(c(2,3,3,4,5,6,10))
dotStraightRanker(c(2,3,4,5,6,7,10))
```

`dotTestDealer` *dotTestDealer*

Description

Assume player 1 already has cards. For remaining players, generate Player+Community cards = 2x(nPlayers-1)+5 cards.

Usage

```
dotTestDealer(nPlayers, position, holeCards)
```

Arguments

nPlayers	number of hands to deal as integer in {2, ... , 9}
position	dealer position as integer in {2, ..., nPlayers}
holeCards	the hand of player 1 as vector[2] in {1, 2, ..., 52}

Value

y : cards dealt in hole as vector[nCards] in {1, 2, ..., 52}

Examples

```
dotTestDealer(9,9,c(1,52))
dotTestDealer(9,5,c(1,52))
dotTestDealer(5,2,c(3,42))
```

---

dotTransformToAbsolute

dotTransformToAbsolute

Description

Transform a relative position (i.e., seats behind the dealer) into an absolute position (i.e., seat at the table).

Usage

```
dotTransformToAbsolute(nPlayers, position, k)
```

Arguments

nPlayers	number of hands to deal as integer in {2, ... , 9}
position	dealer position as integer in {2, ..., nPlayers}
k	relative position of a player as integer in {1, 2, ... , nPlayers}

Value

j : absolute position of a player as integer in {1, 2, ... , nPlayers}

Examples

```
dotTransformToAbsolute(9,9,0)
dotTransformToAbsolute(9,9,8)
dotTransformToAbsolute(9,1,8)
dotTransformToAbsolute(9,5,6)
```

dotTransformToNumber *dotTransformToNumber*

Description

Determine the card from a rank and suit.

Usage

```
dotTransformToNumber(rank, suit)
```

Arguments

rank :

rank of card y as integer in {2, ... , 14}

2 = deuce
.
.
.
11 = jack
12 = queen
13 = king
14 = ace

suit :

suit of card y as integer in {1, 2, 3, 4}

1 = spade
2 = club
3 = heart
4 = diamond

Value

y: number corresponding to card as integer in {1, 2, ... , 52}

Examples

```
dotTransformToNumber(2,1)
dotTransformToNumber(14,1)
dotTransformToNumber(2,2)
dotTransformToNumber(14,2)
```

dotTransformToRank *dotTransformToRank*

Description

Determine the rank of a card.

Usage

```
dotTransformToRank(y)
```

Arguments

y	number corresponding to card as integer in {1, 2, ... , 52}

Value

rank: rank of card y as integer in {2, ... , 14}

2 = deuce
.
.
.
11 = jack
12 = queen
13 = king
14 = ace

Examples

```
dotTransformToRank(1)
dotTransformToRank(13)
dotTransformToRank(14)
dotTransformToRank(26)
```

dotTransformToSuit *dotTransformToSuit*

Description

Determine the suit of a card.

Usage

```
dotTransformToSuit(y)
```

Arguments

y	number corresponding to card as integer in {1, 2, ... , 52}

Value

suit: suit of card y as integer in {1, 2, 3, 4}

1 = spade
2 = club
3 = heart
4 = diamond

Examples

```
dotTransformToSuit(1)
dotTransformToSuit(13)
dotTransformToSuit(14)
dotTransformToSuit(26)
```

---

dotTripRanker *dotTripRanker*

Description

Determine the rank of the three of a kind. Note: dotTripRanker requires a hand with a score of 4 (i.e., three of a kind).

Usage

```
dotTripRanker(oneHand)
```

Arguments

oneHand	:

a sorted hand with ranks only as vector[7]

col1: rank of card 1 in {2, ... , 14}
col2: rank of card 2
.
.
.
col7: rank of card 7

Value

tripRank : the rank of the pair as vector

Examples

```
dotTripRanker(c(9,7,5,3,3,3,2))
```

`dotTrips` *dotTrips*

Description

Determine the player(s) with the highest three of a kind and kicker cards.

Usage

```
dotTrips(nPlayers, cards, score)
```

Arguments

`nPlayers` number of hands as integer in {2, ... , 9}

`cards` :

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

`score` :

the score of the hand in absolute terms as vector[nPlayers]

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

Value

winner : absolute position of the winner as vector

See Also

dotTripRanker and dotHighcardCompare

Examples

```
cards <- c(14,14,4,5,1,2,1,1,10,9,6,7,2,2,2,2,4,4,4,4,3,3,3,3,8,8,8,8,3,3,3,3)
cards <- c(cards,13,13,13,13,3,3,3,3,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
dotTrips(nPlayers,cards,score)

cards <- c(14,14,4,5,1,2,1,1,2,3,6,7,2,2,2,2,4,4,4,4,3,3,3,3,11,11,11,11,3,3,3,3)
cards <- c(cards,13,13,13,13,3,3,3,3,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
dotTrips(nPlayers,cards,score)
```

---

dotTwoPairRanker *dotTwoPairRanker*

Description

Determine the ranks of the two pairs. Notes: dotTwoPairRanker requires a hand with a score of 3 (i.e., two pairs). This functions works best when ranks are sorted in decreasing order.

Usage

```
dotTwoPairRanker(oneHand)
```

Arguments

oneHand :

a sorted hand with ranks only as vector[7]

col1: rank of card 1 in {2, ... , 14}
col2: rank of card 2
.
.
.
col7: rank of card 7

Value

pairRank : the rank of the pair as vector

Examples

```
dotTwoPairRanker(c(9,7,5,3,3,2,2))
dotTwoPairRanker(c(9,5,5,3,3,2,2))
```

`dotTwoPairs` *dotTwoPairs*

Description

Determine the player(s) with the highest two pairs and kicker card.

Usage

```
dotTwoPairs(nPlayers, cards, score)
```

Arguments

`nPlayers` number of hands as integer in {2, ... , 9}

`cards` :

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

`score` :

the score of the hand in absolute terms as vector[nPlayers]

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

Value

winner : absolute position of the winner as vector

See Also

dotTwoPairRanker and dotHighcardCompare

Examples

```
cards <- c(2,3,4,5,1,1,1,1,2,3,6,7,2,2,2,2,4,4,4,4,3,3,3,3,11,11,11,11,3,3,3,3)
cards <- c(cards,13,13,13,13,3,3,3,3,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
dotTwoPairs(nPlayers,cards,score)
```

---

hand *hand*

Description

Assemble the 7 card hands.

Usage

```
hand(players, board)
```

Arguments

players :

the hole cards as matrix[nPlayers, 4]
col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2

board the board cards as vector[5] in {1, 2, ..., 52}

Value

cards : the 7 card hand as matrix[nPlayers, 14]
col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.

.
.
col13: rank of card 7
col14: suit of card 7

See Also

dotTransformToRank and dotTransformToSuit

Examples

```
hand(matrix(1:18,9,2,byrow=TRUE),19:23)
hand(matrix(c(1:9,14:22),9,2),48:52)
```

---

showdown *showdown*

Description

Determine the ranking of the hands.

Usage

```
showdown(cards)
```

Arguments

cards :

the 7 card hand as matrix[nPlayers, 14]
col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2

Value

score : the score of the hand in absolute terms as vector[nPlayers]
9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair

2 = One Pair
1 = High Card

See Also

dotScorer

Examples

```
showdown(matrix( c( 2,1,3,2,5,3,6,4,7,1,13,2,14,2,2,3,2,4,5,1,6,2,7,3,13,4,14,4),2,14,byrow=TRUE))
```

---

testRoundOfPoker *testRoundOfPoker*

Description

Run a test round of poker.

Usage

```
testRoundOfPoker()
```

Value

Outputs a plot window showing the cards dealt as well as prints to the console the summary text, i.e., each hand's score and the winner.

See Also

deal, assignToPlayers, assignToBoard, hand, showdown, tiebreaker, and cgiPlayers

Examples

```
testRoundOfPoker()
```

tiebreaker *tiebreaker*

Description

Determine the winner in the presence of any ties.

Usage

```
tiebreaker(nPlayers, cards, score)
```

Arguments

nPlayers	number of hands as integer in {2, ... , 9}
cards	:

the 7 card hand as matrix[nPlayers, 14]

col1: rank of card 1 in {2, ... , 14}
col2: suit of card 1 in {1, 2, 3, 4}
col3: rank of card 2
col4: suit of card 2
.
.
.
col13: rank of card 7
col14: suit of card 7

score :

the score of the hand in absolute terms as vector[nPlayers]

9 = Straight Flush
8 = Four of a Kind
7 = Full House
6 = Flush
5 = Straight
4 = Three of a Kind
3 = Two Pair
2 = One Pair
1 = High Card

Value

winner : the absolute position of the winner(s) as vector

See Also

dotHighcard, dotPair, dotTwoPairs, dotTrips, dotStraight, dotFlush, dotFullHouse, dotFourOfAKind

and dotStraightFlush

Examples

```
cards <- c(2,1,4,2,5,3,6,4,7,1,13,2,14,3,2,3,3,4,5,1,6,2,7,3,13,4,14,1)
cards <- matrix(cards,2,14,byrow=TRUE); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(2,1,3,2,5,3,6,4,7,1,13,2,14,3,2,3,3,4,5,1,6,2,7,3,13,4,14,1)
cards <- matrix(cards,2,14,byrow=TRUE); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(2,3,4,5,1,1,1,1,2,3,6,7,2,2,2,2,4,4,4,4,3,3,3,3,11,11,11,11,3,3,3,3)
cards <- c(cards,13,13,13,13,3,3,3,3,14,14,14,14,3,3,3,3,9,9,9,9,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(2,3,4,5,1,1,1,1,2,3,6,7,2,2,2,2,4,4,4,4,3,3,3,3,11,11,11,11,3,3,3,3)
cards <- c(cards,13,13,13,13,3,3,3,3,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(14,14,4,5,1,2,1,1,10,9,6,7,2,2,2,2,4,4,4,4,3,3,3,3,8,8,8,8,3,3,3,3)
cards <-c(cards,13,13,13,13,3,3,3,3,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(14,14,4,5,1,2,1,1,2,3,6,7,2,2,2,2,4,4,4,4,3,3,3,3,11,11,11,11,3,3,3,3)
cards <-c(cards,13,13,13,13,3,3,3,3,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(7,1,4,2,4,1,4,3,10,1,11,2,2,2,2,3,3,3,3,3,3,1,1,1,5,5,5,4,4,4,6,6,6)
cards <-c(cards,2,2,2,14,14,14,2,2,2)
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(2,1,4,2,4,1,4,3,10,1,11,2,2,2,2,3,3,3,3,3,3,1,1,1,5,5,5,4,4,4,6,6,6)
cards <-c(cards,2,2,2,14,14,14,2,2,2)
```

```
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(2,1,3,3,5,2,6,3,7,3,13,3,14,3,2,3,3,4,5,1,6,3,7,3,13,3,14,3)
cards <- matrix(cards,2,14,byrow=TRUE); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(2,1,3,3,5,3,6,3,7,3,13,3,14,3,2,3,3,4,5,3,6,3,7,3,13,3,14,3)
cards <- matrix(cards,2,14,byrow=TRUE); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(5,10,4,8,1,2,1,1,10,9,6,7,3,2,2,2,5,5,5,5,3,3,3,3,8,8,8,8,3,3,3,3)
cards <-c(cards,14,14,14,14,2,2,2,2,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(5,10,4,8,1,2,1,1,10,9,6,7,3,2,2,2,12,12,12,12,1,1,1,1,12,12,12,12)
cards <-c(cards,3,3,3,3,14,14,14,14,2,2,2,2,14,14,14,14,3,3,3,3,14,14,14,14,4,4,4,4)
cards <- matrix(cards,nrow=4,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(14,10,5,1,2,1,14,9,7,2,2,2,4,4,4,3,3,3,8,8,8,3,3,3,13,13,13)
cards <-c(cards,3,3,3,14,14,14,3,3,3,14,14,14,4,4,4)
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(3,4,5,1,1,1,8,9,10,1,1,1,14,14,14,1,1,1,14,14,14,2,2,2,11,11,11)
cards <-c(cards,3,3,3,14,14,14,3,3,3,14,14,14,4,4,4)
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(8,13,5,1,1,4,6,2,2,2,3,4,14,14,14,2,2,2,9,9,9,1,1,1,10,10,10)
cards <-c(cards,1,1,1,11,11,11,1,1,1,12,12,12,1,1,1)
cards <- matrix(cards,nrow=3,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)

cards <- c(1,1,3,4,2,2,3,4,8,8,1,1,9,9,1,1,10,10,1,1,11,11,1,1,12,12,1,1)
```

```
cards <- matrix(cards,nrow=2,ncol=14); cards
score <- showdown(cards); score
nPlayers <- nrow(cards); nPlayers
tiebreaker(nPlayers,cards,score)
```

`transformToRelative` *transformToRelative*

Description

Transforms an absolute position (i.e., seat at the table) into a relative position (i.e., seats behind the dealer)

Usage

```
transformToRelative(nPlayers, position, j)
```

Arguments

`nPlayers`	number of hands to deal as integer in {2, ... , 9}
`position`	dealer position as integer in {2, ..., nPlayers}
`j`	absolute position of a player as integer in {1, 2, ... , nPlayers}

Value

k : relative position of a player as integer in {1, 2, ... , nPlayers}

Examples

```
transformToRelative(9,9,9)
transformToRelative(9,9,8)
transformToRelative(9,1,9)
transformToRelative(9,5,2)
```

Index