

# Package ‘fourinarow’

May 8, 2026

**Type** Package

**Title** Play ‘‘Four in a Row’’

**Version** 0.1.1

**Description** Play or simulate games of ‘‘Four in a Row’’ in the R console. This package is designed for educational purposes, encouraging users to write their own functions to play the game automatically. It contains a collection of built-in functions that play the game at various skill levels, for users to test their own functions against.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Suggests** knitr, quarto

**VignetteBuilder** quarto

**Depends** R (>= 3.5)

**Imports** methods

**NeedsCompilation** no

**Author** Kelly Street [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-6379-5013>>)

**Maintainer** Kelly Street <street.kelly@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-11-21 11:40:02 UTC

## Contents

bots . . . . .	2
getSetsof4 . . . . .	3
humanPlayer . . . . .	3
invertPieces . . . . .	4
play4inaRow . . . . .	4
testBots . . . . .	5
<b>Index</b>	<b>7</b>

---

bots

*Bot players for Four in a Row*

---

### **Description**

Computer players that select their next move based on various amounts of internal logic.

### **Usage**

`randomBot(game)`

`easyBot(game)`

`mediumBot(game)`

`hardBot(game)`

### **Arguments**

`game`                    A 6x7 matrix object representing the current game board.

### **Value**

Returns an integer between 1 and 7. Each bot only selects from the set of valid moves, so they won't select a column that is already full.

### **Functions**

- `randomBot()`: Chooses moves randomly.
- `easyBot()`: Tries to make 4 in a row, but does not consider its opponents moves.
- `mediumBot()`: Selects a move based on simple internal logic. It tries to make 4 in a row and tries to block the opponent from winning, but does not consider possible downstream moves.
- `hardBot()`: Selects a move by looking three moves ahead (with downstream moves selected by internal logic similar to `mediumBot`).

### **Examples**

```
play4inaRow(randomBot, easyBot)
```

```
play4inaRow(mediumBot, hardBot)
```

---

getSetsof4	<i>All possible sets of 4 in a row</i>
------------	--

---

**Description**

Manually constructs a dataset containing all possible winning sets of four in a standard (6x7) game of Four in a Row.

**Usage**

```
getSetsof4()
```

**Value**

A matrix with 69 rows and 4 columns. Each row contains numeric indices for a potential winning set of four in a row. For example, sets[1, ] represents the set of four vertical spaces starting in the top left corner.

**Examples**

```
getSetsof4()
```

---

humanPlayer	<i>Human player</i>
-------------	---------------------

---

**Description**

Get moves as input from the command line, allowing users to play against a bot.

**Usage**

```
humanPlayer(game)
```

**Arguments**

game	A 6x7 matrix object representing the current game board.
------	--

**Details**

While possible, human vs. human games can be confusing because the game switches Xs and Os between turns (so that every player sees their own pieces as X).

**Value**

Prints the current game board and prompts the user to input a move, which must be an integer between 1 and 7 and a valid move in the current game.

**Examples**

```
play4inaRow(humanPlayer, randomBot)
```

---

invertPieces	<i>Switch X's and O's</i>
--------------	---------------------------

---

**Description**

Convenience function called internally by play4inaRow that switches the player symbols, so that each player sees their own pieces as "X" and their opponents' as "O".

**Usage**

```
invertPieces(game)
```

**Arguments**

game                    A 6x7 matrix object representing the current game board.

**Value**

Returns the game matrix with "X" and "O" symbols switched.

**Examples**

```
game <- matrix(sample(c('X','O','.'), 6*7, replace=TRUE), nrow = 6, ncol = 7)
game
invertPieces(game)
```

---

play4inaRow	<i>Play a game of Four in a Row</i>
-------------	-------------------------------------

---

**Description**

Play a game of Four in a Row

**Usage**

```
play4inaRow(playerOne, playerTwo, verbose = TRUE)
```

**Arguments**

playerOne	A function that takes the current board as input and returns the next move (1-7) for Player 1.
playerTwo	Same for Player 2. Note that both functions see their pieces as "X" and opponent's pieces as "O".
verbose	Logical value indicating whether or not to print the final board to the console (default is TRUE)

**Details**

The game is played on a  $6 \times 7$  grid and players alternate placing markers in one of the 7 columns. The piece will "fall" to the lowest unoccupied space in that column. The game ends when one player wins by getting four pieces in a row (horizontally, vertically, or diagonally).

Note that every player will see their markers as Xs when it is their turn.

**Value**

Returns 1 or 2 to indicate whether Player 1 or Player 2 was the winner. Returns 0 in the case of a tie.

**Examples**

```
play4inaRow(randomBot, randomBot)
```

---

testBots	<i>Simulate many games of Four in a Row</i>
----------	---

---

**Description**

This is a convenient way to test two bots against each other over a large number of games while ensuring that neither bot has an advantage from going first more often than the other.

**Usage**

```
testBots(playerOne, playerTwo, n = 100)
```

**Arguments**

playerOne	A function that takes the current board as input and returns the next move (1-7) for Player 1.
playerTwo	Same for Player 2. Note that both functions see their pieces as "X" and opponent's pieces as "O".
n	Numerical value of how many games to simulate. Must be an even number, so that both bots get the same number of games with the first move (default = 100).

**Value**

Returns a vector of counts for the number of ties, wins by playerOne, and wins by playerTwo.

**Examples**

```
testBots(easyBot, randomBot)
```

# Index

`bots`, [2](#)

`easyBot (bots)`, [2](#)

`getSetsof4`, [3](#)

`hardBot (bots)`, [2](#)

`humanPlayer`, [3](#)

`invertPieces`, [4](#)

`mediumBot (bots)`, [2](#)

`play4inaRow`, [4](#)

`randomBot (bots)`, [2](#)

`testBots`, [5](#)