How this game works

Thank you for your curiosity into how this game works!

The motivation behind creating this game was to introduce R in a fun way to those who are interested in coding or who may rely on R or statistical programming languages in the future. It is supposed to be easy to play for R beginners while also allowing players to go into the source code directly to see specifically how this program runs. This game touches upon conditional statements, matrix operations, and plotting simple shapes. Keep in mind there are many ways one could create this game in R; here, I present what made the most sense to me!

This document broadly outlines the structure of the R code and the logic R follows to play Tic-Tac-Toe. To understand the details of the game, you can read the R source code (in the "code" folder) or contacting me (chris.j.blackford@gmail.com or github.com/Christopher-Blackford). You may use this code for any projects you like but please contact me if you do so I know it has been useful!

Visualizing Tic-Tac-Toe

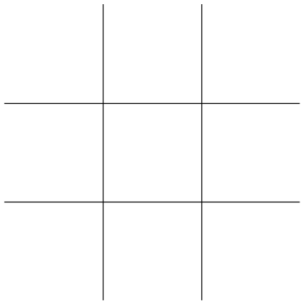
Plotting custom shapes in R can be tricky, so the visualization of the Tic-Tac-Toe board was done by using Rs ability to visualize spatial data. This is why this game requires the *sp* and *rgeos* packages. The Xs, Os, and game board are all stored as spatial line data (think drawing on graph paper). All possible X and O positions are generated at the beginning of the game, and a subset of those are displayed in the plot window, depending on the moves the player and computer make.

How R/Computer understands player input

The player's input is stored in R in 2 ways. First, it is stored as a visualization on the game board that can be seen in the plotting window. It is also stored, "behind the scenes", in a 3x3 matrix that mimics the game board.

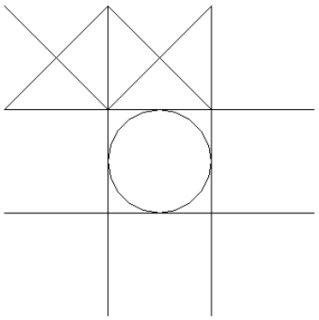
At the beginning of the game, this 3x3 matrix is filled with zeros. Each time the player moves, it replaces one of these zeros with a 1. Each time the computer moves, it replaces one of these zeros with a 10. The replacement occurs in the same location on the game board as in the matrix:

```
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
```



At beginning of game, both board and matrix are empty

```
     [,1] [,2] [,3]
[1,]    1    1    0
[2,]    0   10    0
[3,]    0    0    0
```



Example of the game board and matrix after 3 turns

By summing the rows, columns, and diagonals of the matrix, the computer can infer the positions of the Xs and Os across the board.

How R/Computer plays against the player

Tic-Tac-Toe is a simple enough game that, by following a few rules, the computer can ensure it will never lose a game.

First move.

The first computer move is coded different from the subsequent moves to respond optimally to the opening player move. If the player moves into any corner space, the computer moves into the centre. If the player moves into the centre, the computer moves into one of the four corners. If the player moves into any of the other spaces (position 2,4,6, or 8), the computer moves into one of the adjacent corners.

Subsequent moves.

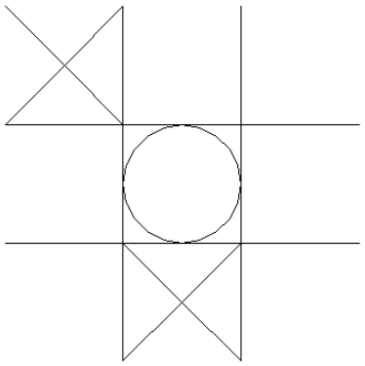
After the first move, the rest of the computer moves follow the same procedure:

1. Computer sums rows, columns, and diagonals of matrix to determine location of Xs and Os

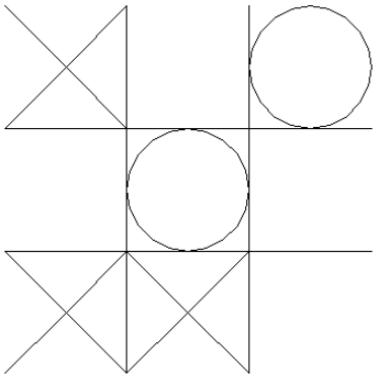
2. The computer is in an “offensive mode”. If the computer can win in one move, computer will place an O to win the game. If the computer can’t win in one move, go onto step 3

3. The computer is in a “defensive mode”. If the player can win in one move, computer will place an O to block the player. If the player can’t win, go onto step 4.

4. The computer is in a “defensive mode”. There are some situations where a player can’t win in one move but if the computer plays suboptimally, the player is guaranteed a win in 2 turns. The computer checks for these situations and plays to counter them. One example of a “dangerous” board is shown below:



At first glance, it is tempting for the computer to move into the top-right corner as it will put it one move away from winning. However, if computer goes top-right, the player can counter and then is set up to win regardless of the computers next move:



If these “dangerous” board aren’t detected, go onto step 5.

5. The computer is in an “offensive mode”. If neither the computer or the player are one move away from winning, and these “dangerous” boards aren’t detected, the computer plays an O in an open row/column/diagonal such that it is now one move away from winning. In practice, the computer rarely gets to step 5 but if the player is playing poorly it can occur.

Difficulty settings

The Impossible difficulty setting follows the above rules perfectly. The Easy, Medium, and Hard modes are essentially dumbed down versions of the Impossible difficulty. These modes follow the same rules as above most of the time, but in each step incorporate a chance of moving randomly instead of following the rule. As the difficulty decreases, the chance of moving randomly increases.