QGameTheory

Indranil Ghosh
School of Fundamental Sciences
Massey University
Manawatu, Private Bag 11 222, Palmerston North 4442, New Zealand
i.ghosh@massey.ac.nz

Abstract

*QGameTheory is a contributed **R** package that helps in simulating quantum versions of various game theory models. It is a general-purpose toolbox that works with a simple quantum computing framework, known as the quantum circuit model to perform various computations. The current version of the package makes use of a maximum of six qubits and a total number of seven game theory models. Application of quantum computation in game theory, which gives rise to many interesting results, like the escape from the dilemma in the case of Quantum Prisoner's Dilemma, which otherwise persists in the classical case. The objective of this presentation is to introduce the package to interested students starting in this field, using which they can learn quantum computation and game theory concepts. Quantum versions of models that have been handled within the package are: **Penny Flip Game, Prisoner's Dilemma, Two Person Duel, Battle of the Sexes, Hawk and Dove Game, Newcomb's Paradox** and **Monty Hall Problem**.*

1. Package Logo

2. Repository

The development version can be downloaded from the github repository:

```
> devtools::install_github("indrag49/QGameTheory")
```

It can also be downloaded from the **CRAN** repository by:

```
> install.packages("QGameTheory")
```

After downloading the package, activate the package and initialize all the variables inside it, using the following command:

```
> init()
```

3. Introduction

A quantum environment has been already defined inside the package:

```
Q <<- new.env(parent=emptyenv())
```

the user can define qubits, qutrits and any possible quantum states to begin with. Qubit $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ can be initialized in the following way:

```
> Q$Q0
     [,1]
[1,]    1
[2,]    0
```

Similarly, qubit $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ can be defined too. An arbitrary quantum state, for example, $|\psi\rangle = |0\rangle \otimes (\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)$, can be initialized using the package:

```
> psi = kronecker(Q$Q0, (Q$Q0/sqrt(2) + Q$Q1/sqrt(2)))
```

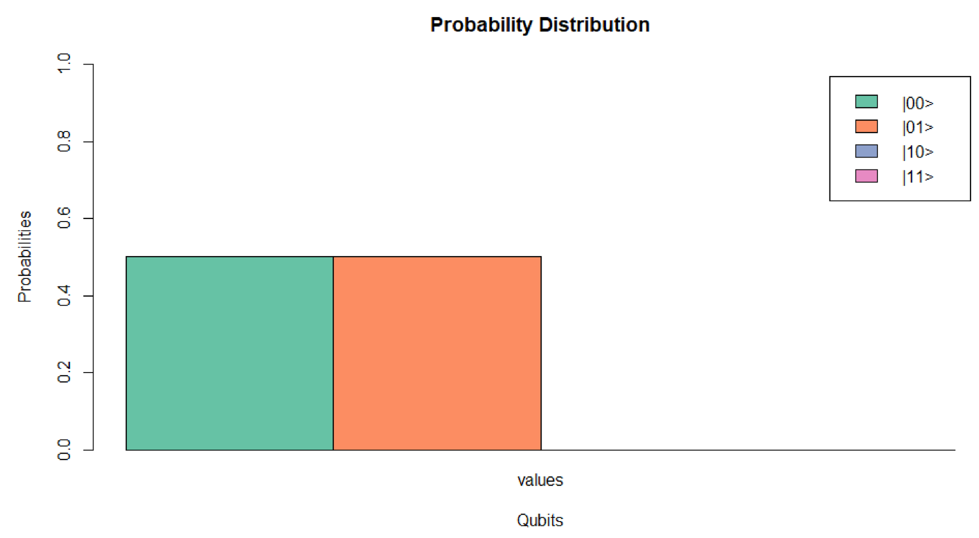
The vector form of $|\psi\rangle$ is:

```
> psi
          [,1]
[1,] 0.7071068
[2,] 0.7071068
[3,] 0.0000000
[4,] 0.0000000
```

One can also perform measurement of the quantum state, using the command:

```
> QMeasure(psi)
```

The command will yield us the following probability distribution plot:



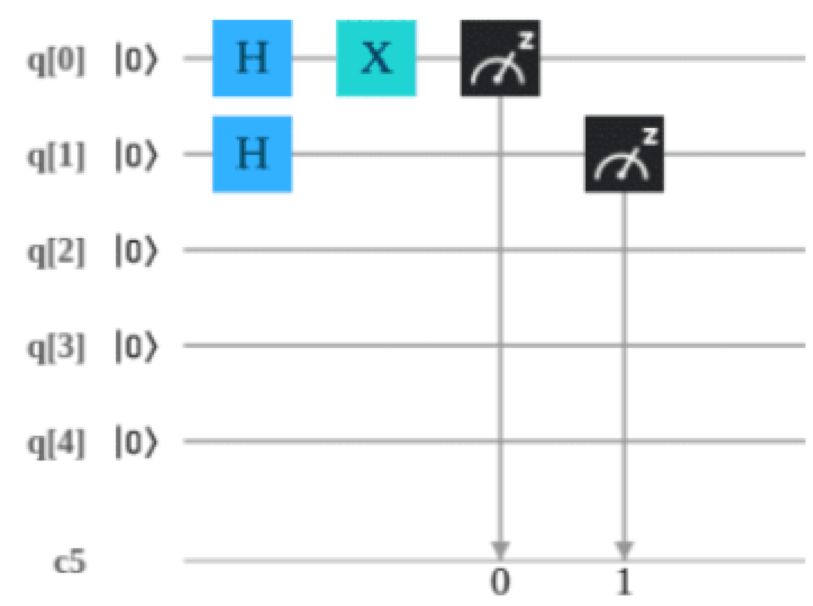
The package allows us to work with *quantum logic gates* too, that are unitary matrices and help us simulate quantum algorithms. Some of them are the identity operator , the Pauli gates, the Hadamard gate, CNOT, Fredkin, T, Toffoli, and more, along with the Gell Mann matrices representing 3-system logic gates. For example, the **Hadamard** gate $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ can be written as:

```
> Hadamard(Q$I2)
          [,1]       [,2]
[1,] 0.7071068  0.7071068
[2,] 0.7071068 -0.7071068
```

One can define the **Bell** states and also perform **Quantum Fourier Transform** with the package

4. A Quantum Algorithm

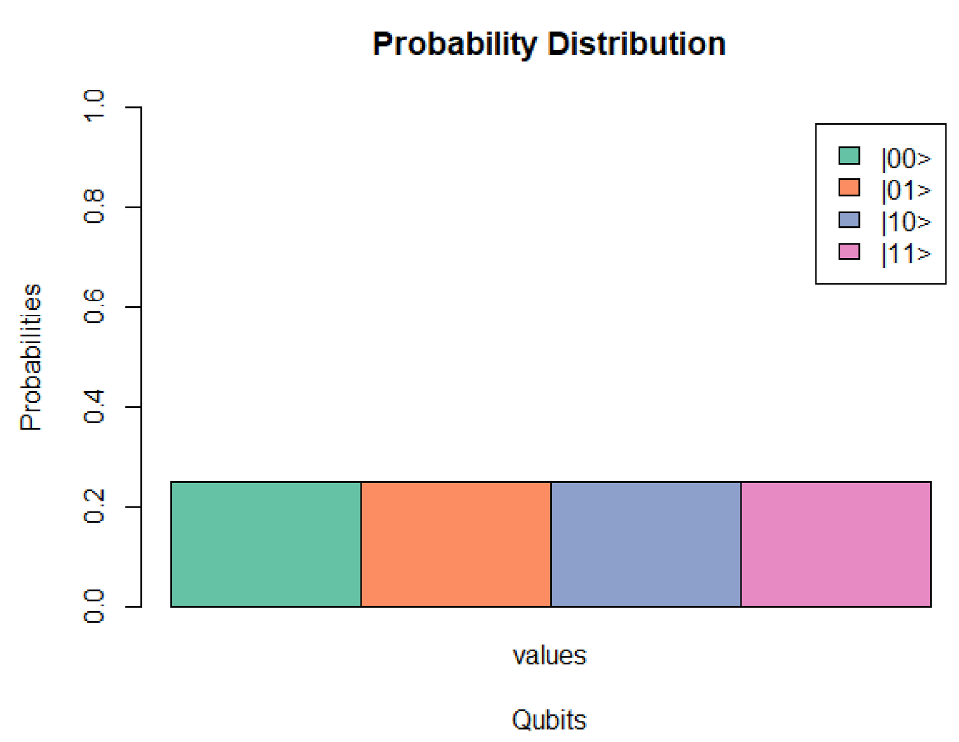
Let a simple quantum algorithm (generated with IBM Q Experience) look like:



Using **QGameTheory** we can write an **R** script to simulate the same:

```
Psi <- Q$Q00 # initialize the quantum state
H <- Hadamard(Q$I2) # Hadamard Gate, I2 is the identity
  operator
X <- sigmaX(Q$I2) # Pauli-X Gate
HH <- kronecker(H, H) # outer product between two Hadamard
  Gates
XI <- kronecker(X, Q$I2) # outer product between Hadamard and
  identity
Psi1 <- HH %*% Psi # intermediate quantum state
Psif <- XI %*% Psi1 # final quantum state
QMeasure(Psif) # measure the quantum state
```

The probability distribution plot after measurement:



5. Game Theory Concepts

The **QGameTheory** package allows us to work with the **Iterated Deletion of Strictly Dominated Strategies (IDSDS) algorithm** and also provides us with functionality to calculate **NASH equilibrium** from payoff matrices. For example:

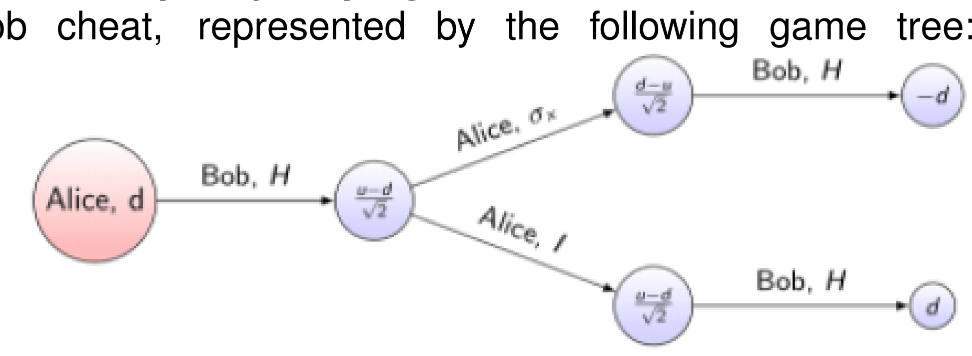
```
> A <- matrix(c(10, 5, 3, 0, 4, 6, 2, 3, 2), ncol=3, byrow=
  TRUE)
> B <- matrix(c(4, 3, 2, 1, 6, 0, 1, 5, 8), ncol=3, byrow=TRUE
  )
> IDSDS(A, B)
[[1]]
     [,1]
[1,]   10

[[2]]
     [,1]
[1,]    4
```

The `IDSDS()` function gives the indices along with the corresponding values of the strictly dominant strategies from the payoff matrices. The `NASH()` function gives only the indices, as illustrated below:

```
> NASH(A, B)
Joining, by = c("V1", "V2")
  V1 V2
1  1  1
```

6. Quantum Penny Flip Game

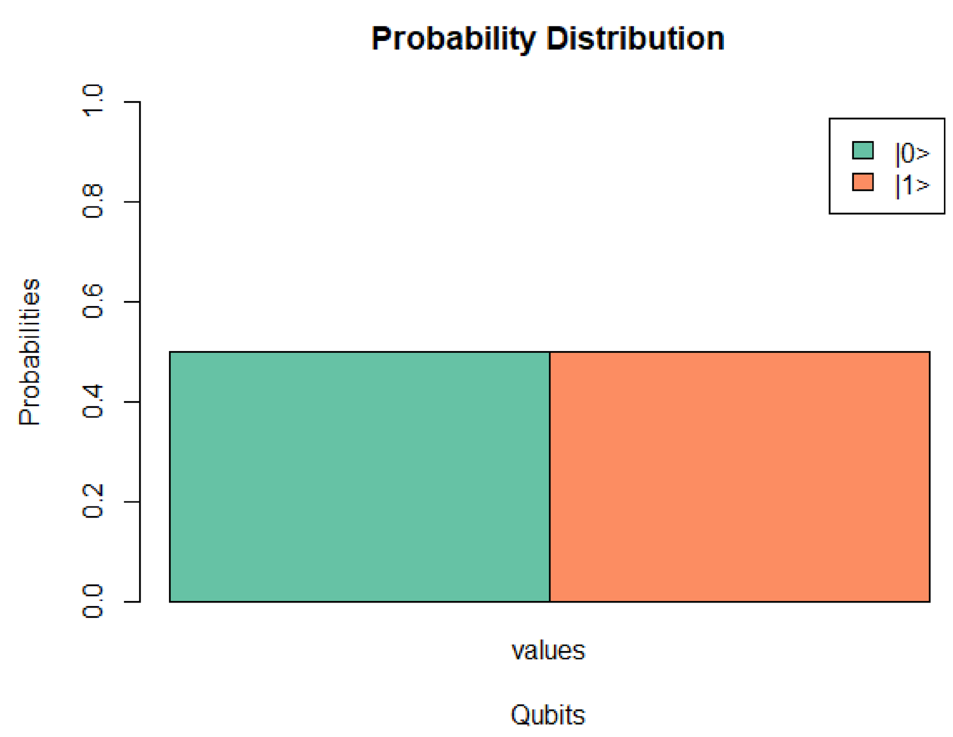
We consider, for example, a particular case of the **quantum penny flip game** where both Alice and Bob cheat, represented by the following game tree:



The **R** script simulating the above case:

```
Psi <- (Q$Q0+Q$Q1)/sqrt(2)
S1 <- sigmaX(Q$I2)
S2 <- Q$I2
H <- Hadamard(Q$I2)
SA <- list(S1, S2)
SB <- list(H)
QPennyFlip(Psi, SA, SB)
```

The above code will also generate the following probability distribution plot simulating the result that helps us in our analysis:



We see that both Alice and Bob have the equal probability of winning the game.

7. Conclusion

The **QGameTheory** package provides us with other functionalities to analyse the remaining six quantum game models described before. It also gives us access to miscellaneous functions, for example, the `levi_civita()` function that might be required for our analysis.

8. Disclaimer

It is to inform the readers that the views, thoughts, and opinions expressed herein, belong solely to the author, and not to the author's employer, organization or any other group or individual.

References

[1] I. Ghosh. QGameTheory: Quantum Game Theory Simulator (v0.1.2). *CRAN*, 2020.

[2] M. Nielsen and I. Chuang. Quantum Computation and Quantum Information. *ISBN-978-1-107-00217-3*, 2010.

[3] J. O. Grabbe. An Introduction to Quantum Game Theory. *arXiv:quant-ph/0506219*, 2005.

[4] LaTeX template derived from the file 'sciposter.cls' v 1.18 authors Michael H.F. Wilkinson and Alle Meije Wink, August 18, 2006.