Package: qvirus (via r-universe)

August 27, 2024

Title Quantum Computing for Analyzing CD4 Lymphocytes and Antiretroviral Therapy

Version 0.0.2

Description Resources, tutorials, and code snippets dedicated to exploring the intersection of quantum computing and artificial intelligence (AI) in the context of analyzing Cluster of Differentiation 4 (CD4) lymphocytes and optimizing antiretroviral therapy (ART) for human immunodeficiency virus (HIV). With the emergence of quantum artificial intelligence and the development of small-scale quantum computers, there's an unprecedented opportunity to revolutionize the understanding of HIV dynamics and treatment strategies. This project leverages the R package 'qsimulatR' (Ostmeyer and Urbach, 2023, <https://CRAN.R-project.org/package=qsimulatR>), a quantum computer simulator, to explore these applications in quantum computing techniques, addressing the challenges in studying CD4 lymphocytes and enhancing ART efficacy.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests bookdown, knitr, qsimulatR, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/juanv66x/qvirus

BugReports https://github.com/juanv66x/qvirus/issues

Repository https://juanv66x.r-universe.dev

RemoteUrl https://github.com/juanv66x/qvirus

RemoteRef HEAD

RemoteSha 54a258c594cc0297fecc006eabed0e33aef94c5f

Contents

`complex_check` *Check if Coefficients of a Qubit State Object are Complex Numbers*

Description

This function returns the class of the coefficients of a given qubit state object

Usage

```
complex_check(qstate_obj)
```

Arguments

`qstate_obj`	A qubit state object created using the qsimulatR package.

Value

Coefficients class of given qubit state object

Examples

```
library(qsimulatR)
ket1 <- qstate(nbits = 1, coefs = c(0, 1))
complex_check(ket1)
```

conjugate_transpose *Calculate the Conjugate Transpose of a Quantum State*

Description

This function calculates the conjugate transpose of a given quantum state represented by a qstate object.

Usage

```
conjugate_transpose(state)
```

Arguments

state	A qstate object for which the conjugate transpose is to be calculated.

Value

The conjugate transpose of the input quantum state.

Examples

```
library(qsimulatR)
library(qvirus)
# Calculate the conjugate transpose of ket0
state <- six_state(1)[[1]]
conjugate_transpose(state)
```

normalize_check *Normalize Check Function for qstate Class Object*

Description

Check the normalization of a qstate object created by qsimulatR package.

Usage

```
normalize_check(qstate_obj, probs = FALSE)
```

Arguments

qstate_obj	A quantum state object.
probs	Are probabilities required as output?

Value

Either the sum of the squared magnitudes of the coefficients of the qstate object or its probabilities.

Examples

```
library(qsimulatR)
ket0 <- qstate(nbits = 1)
normalize_check(ket0)
```

---

phen_hiv *Calculate Final State and Payoffs in Quantum Game*

Description

This function calculates the final quantum state and expected payoffs for two players in a quantum game based on their strategies. The function uses quantum gates and unitary transformations to simulate the game dynamics.

Usage

```
phen_hiv(strategy1, strategy2, alpha, beta, gamma, theta)
```

Arguments

strategy1	A 2x2 matrix representing the strategy of player 1.
strategy2	A 2x2 matrix representing the strategy of player 2.
alpha	A numeric value representing the payoff for outcome |00>.
beta	A numeric value representing the payoff for outcome |01>.
gamma	A numeric value representing the payoff for outcome |10>.
theta	A numeric value representing the payoff for outcome |11>.

Value

A list containing the final quantum state (`final_state`), the payoffs for each basis state (`payoffs`), and the expected payoffs for player 1 (`pi_v`) and player 2 (`pi_V`).

References

Özlüer Başer, B. (2022). "Analyzing the competition of HIV-1 phenotypes with quantum game theory". Gazi University Journal of Science, 35(3), 1190–1198. doi:10.35378/gujs.772616

Examples

```
library(qsimulatR)
strategy1 <- diag(2) # Identity matrix for strategy 1
strategy2 <- diag(2) # Identity matrix for strategy 2
alpha <- 1
beta <- 0.5
gamma <- 2
theta <- 0.1
result <- phen_hiv(strategy1, strategy2, alpha, beta, gamma, theta)
print(result)
```

pure_qubit1 *Create a normalized pure quantum state for a 1-qubit system.*

Description

Create a normalized pure quantum state for a 1-qubit system.

Usage

```
pure_qubit1(theta, phi, spherical = FALSE)
```

Arguments

theta	The parameter theta in radians.
phi	The parameter phi in radians.
spherical	Whether to return coordinates in spherical form (default is FALSE).

Value

A qstate object representing the normalized pure quantum state for a 1-qubit system.

Examples

```
# Quantum simulator
library(qsimulatR)
# Define the parameters
theta <- pi/4
phi <- pi/6
# Create the quantum state
psi_qubit1 <- pure_qubit1(theta, phi)
psi_qubit1
```

pure_qubit2 *Create a normalized pure quantum state for a 2-qubit system.*

Description

Create a normalized pure quantum state for a 2-qubit system.

Usage

```
pure_qubit2(theta1, theta2, phi1, phi2)
```

Arguments

theta1	The parameter theta1 in radians for the first qubit.
theta2	The parameter theta2 in radians for the second qubit.
phi1	The phase parameter phi1 in radians for the first qubit.
phi2	The phase parameter phi2 in radians for the second qubit.

Value

A qstate object representing the normalized pure quantum state for a 2-qubit system.

Examples

```
#'
# Quantum simulator
library(qsimulatR)
# Define the parameters
theta1 <- pi/3
theta2 <- pi/4
phi1 <- pi/6
phi2 <- pi/5

# Create the quantum state
psi_qubit2 <- pure_qubit2(theta1, theta2, phi1, phi2)
psi_qubit2
```

---

simulate_entanglement *Simulate Entanglement Evolution*

Description

This function simulates the evolution of entanglement between two quantum states x1 and x2 using the CNOT gate.

Usage

```
simulate_entanglement(x1, x2, iterations, angle, verbose = FALSE)
```

Arguments

x1	Quantum state for qubit 1, represented as a qstate object.
x2	Quantum state for qubit 2, represented as a qstate object.
iterations	Number of iterations for the entanglement process.
angle	Rotation angle for applying Rx gate.
verbose	If TRUE, prints detailed information to the console.

Value

A list containing the entangled quantum state x2 after each iteration and other relevant information.

Examples

```
library(qsimulatR)
library(qvirus)
x1 <- qstate(1, coefs = as.complex(c(0.8, 0.6)))
x2 <- qstate(1, coefs = as.complex(c(0.38, 0.92)))
results <- simulate_entanglement(x1, x2, iterations = 3, angle = pi/4, verbose = TRUE)
print(results)
```

---

six_state *Create Six Important States on the Bloch Sphere*

Description

This function creates and returns six important states on the Bloch Sphere based on the specified vector numbers.

Usage

```
six_state(vec_num = c(1, 2, 3, 4, 5, 6))
```

Arguments

vec_num	A numeric vector specifying the indices of the states to include. Valid indices are 1 to 6.

Value

A list containing the selected quantum states based on the input vector vec_num.

Examples

```
library(qsimulatR)
# Select and return states 1, 3, and 5
six_state(c(1, 3, 5))
```

Index