Package ‘qsimulatR’

December 9, 2020

Version 1.0

Date 2020-12-07

Title A Quantum Computer Simulator

Description A quantum computer simulator framework with up to 24 qubits. It allows to define general single qubit gates and general controlled single qubit gates. For convenience, it currently provides the most common gates (X, Y, Z, H, S, T, Rx, Ry, Rz, CNOT, SWAP, Toffoli or CCNOT, Fredkin or CSWAP). ‘qsimulatR’ supports plotting of circuits and is able to export circuits to ‘Qiskit’<https://qiskit.org/>>, a python package which can be used to run on IBM’s hardware <https://quantum-computing.ibm.com/>.

Imports methods, stats

Suggests knitr, markdown, rmarkdown

License GPL-3

LazyData true

RoxygenNote 7.1.1

Encoding UTF-8

VignetteBuilder knitr

URL https://github.com/HISKP-LQCD/qsimulatR

BugReports https://github.com/HISKP-LQCD/qsimulatR/issues

Collate 'state.R' 'sqgate.R' 'ccqgate.R' 'cnotgate.R' 'cqgate.R' 'export2qiskit.R' 'measure.R' 'phase_estimation.R' 'plot-qstate.R' 'qft.R' 'qsimulatR-package.R' 'swapgate.R'

NeedsCompilation no

Author Johann Ostmeyer [aut], Carsten Urbach [aut, cre]

Maintainer Carsten Urbach <urbach@hiskp.uni-bonn.de>

Repository CRAN

Date/Publication 2020-12-09 09:20:05 UTC
### R topics documented:

* .cnotgate.qstate-method .......................... 3
* .ccnotgate.qstate-method ...................... 3
* .cnotgate.qstate-method ....................... 4
* .complex.qstate-method ......................... 4
* .cqgate.qstate-method ........................... 5
* .cswapgate.qstate-method ....................... 5
* .matrix.qstate-method ........................... 6
* .sqgate.qstate-method ........................... 6
* .swapgate.qstate-method ......................... 7
CCNOT ............................................. 7
ccnotgate .......................................... 8
ccqgate ........................................... 8
CNOT ................................................ 9
cnotgate ............................................. 9
cqft .................................................. 10
cqgate .............................................. 11
CSWAP ............................................... 11
cswapgate ........................................... 12
export2qiskit ....................................... 12
genComputationalBasis ............................. 13
genStateNumber ..................................... 14
genStateString ...................................... 14
H ..................................................... 15
hist.measurement ................................... 15
Id ..................................................... 16
is.bitset ........................................... 17
measure .............................................. 17
normalise ........................................... 18
phase_estimation ................................... 18
plot.qstate,missing-method ...................... 19
qft .................................................... 20
qsimulatR ........................................... 21
qstate .............................................. 21
Ri ..................................................... 22
Rx ..................................................... 23
Ry ..................................................... 23
Rz ..................................................... 24
S ....................................................... 24
sqgate ............................................... 25
summary.measurement ................................ 26
SWAP ................................................. 26
swapgate ............................................ 27
Tgate ................................................ 27
truth.table ......................................... 28
X ..................................................... 29
Y ..................................................... 29
Description
Applies a CCNOT (or toffoli) gate to a quantum state.

Usage
```r
## S4 method for signature 'ccnotgate,qstate'
e1 * e2
```

Arguments
- `e1` object of S4 class `ccnotgate`
- `e2` object of S4 class `qstate`

Value
An object of S4 class `qstate`

Description
applies a twice controlled single qubit gate to a quantum state.

Usage
```r
## S4 method for signature 'ccqgate,qstate'
e1 * e2
```

Arguments
- `e1` object of S4 class `ccqgate`
- `e2` object of S4 class `qstate`

Value
An object of S4 class `qstate`
**Description**

Applies a CNOT gate to a quantum state.

**Usage**

```r
## S4 method for signature 'cnotgate,qstate'
e1 * e2
```

**Arguments**

- `e1`: object of S4 class `cnotgate`
- `e2`: object of S4 class `qstate`

**Value**

An object of S4 class `qstate`

---

**Description**

Multiplies a quantum gate by a global (phase) factor.

**Usage**

```r
## S4 method for signature 'complex,qstate'
e1 * e2
```

**Arguments**

- `e1`: object of S4 class `complex`
- `e2`: object of S4 class `qstate`

**Value**

An object of S4 class `qstate`
**times-cqgate-qstate**

**Description**
Applies a controlled single qubit gate to a quantum state.

**Usage**
```r
## S4 method for signature 'cqgate,qstate'
e1 * e2
```

**Arguments**
- `e1` object of S4 class `cqgate`
- `e2` object of S4 class `qstate`

**Value**
An object of S4 class `qstate`

**times-cswapgate-qstate**

**Description**
Applies a CSWAP gate to a quantum state.

**Usage**
```r
## S4 method for signature 'cswapgate,qstate'
e1 * e2
```

**Arguments**
- `e1` object of S4 class `cswapgate`
- `e2` object of S4 class `qstate`

**Value**
An object of S4 class `qstate`
Description
Applies a single qubit gate to a quantum state.

Usage
## S4 method for signature 'matrix,qstate'
e1 * e2

Arguments
  e1          object of S4 class 'matrix'
e2          object of S4 class 'qstate'

Value
An object of S4 class 'qstate'

Description
Applies a single qubit gate to a quantum state.

Usage
## S4 method for signature 'sqgate,qstate'
e1 * e2

Arguments
  e1          object of S4 class 'sqgate'
e2          object of S4 class 'qstate'

Value
An object of S4 class 'qstate'
### Description

Applies a SWAP gate to a quantum state.

#### Usage

```r
## S4 method for signature 'swapgate,qstate'
e1 * e2
```

#### Arguments

- `e1`: object of S4 class `swapgate`
- `e2`: object of S4 class `qstate`

#### Value

An object of S4 class `qstate`

---

### CCNOT

The CCNOT or toffoli gate

#### Description

The CCNOT or toffoli gate

#### Usage

```r
CCNOT(bits = c(1, 2, 3))
toffoli(bits = c(1, 2, 3))
```

#### Arguments

- `bits`: integer vector of length two, the first bit being the control and the second the target bit.

#### Value

An S4 class `ccnotgate` object is returned
ccnotgate

The CCNOT gate

Description

This class represents a generic CNOT gate

Slots

bits  Integer vector of length 2. First two bits are the control bits, third the target bit.

Examples

```r
x <- qstate(nbits=3)
z <- CCNOT(c(1,2,3)) * (H(1) * x)
```

ccqgate

A twice controlled single qubit gate

Description

This class represents a generic controlled gate

Details

The qubits are counted from 1 to nbits starting with the least significant bit.

Slots

bits  Integer. Integer vector of bits. The first two are the control bits, the third the target bit.
gate  sqgate. The single qubit gate.

Examples

```r
x <- H(1) * qstate(nbits=3)
## application of the CCX (CCNOT) gate to bit 1,2,3
z <- ccqgate(bits=c(1L, 2L, 3L), gate=X(3L)) * x
z
## the same, but differently implemented
z <- CCNOT(c(1,2,3)) * x
z
```
The CNOT gate

Description
The CNOT gate

Usage
CNOT(bites = c(1, 2))

Arguments
bits integer vector of length two, the first bit being the control and the second the target bit.

Value
An S4 class 'cnotgate' object is returned

cnotgate The CNOT gate

Description
This class represents a generic CNOT gate

Slots
bits Integer vector of length 2. First bit is the control bit, second the target bit.

Examples
x <- qstate(nbits=2)
## A Bell state
z <- CNOT(c(1,2)) * (H(1) * x)
cqft

description

performs the controlled quantum Fourier Trafo on the qstate x and the specified list of qubits.

Usage

cqft(c, x, inverse = FALSE, bits)

Arguments

c integer. a single control qubit.
x qstate. state the qft will applied to
inverse boolean. If 'TRUE', perform inverse transform
bits integer. list of qubits to include in the trafo. if missing, bits=c(1:n)[-c] is assumed, with n the number of qubits in x.

Details

Controlled Quantum Fourier Trafo

The Fourier Trafo is defined as

\[ |j > = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp(2\pi i j k/N)|k > \]

the inverse with the opposite sign in the exponential.

Value

a qstate object with the quantum Fourier trafo of input x.

Examples

x <- qstate(3)
y <- cqft(1, x)
z <- cqft(1, y, inverse=TRUE)
cqgate

A controlled single qubit gate

Description
This class represents a generic controlled gate

Details
The qubits are counted from 1 to nbits starting with the least significant bit.

Slots

bits Integer. Integer vector of bits. The first is the control bit, the second the target bit.
gate sqgate. The single qubit gate.

Examples

```r
x <- H(1) * qstate(nbits=2)
## application of the CX (CNOT) gate to bit 1,2
z <- cqgate(bits=c(1L, 2L), gate=X(2L)) * x
z
## the same as, but differently implemented
z <- CNOT(c(1,2)) * x
z
```

CSWAP

The CSWAP or Fredkin gate

Description
The CSWAP or Fredkin gate

Usage

CSWAP(bits = c(1, 2, 3))

fredkin(bits = c(1, 2, 3))

Arguments

bits integer vector of length two, the first bit being the control and the second the target bit.

Value
An S4 class 'cswapgate' object is returned
The CSWAP gate

Description

This class represents a generic SWAP gate, also called Fredkin gate

Slots

bits Integer vector of length 2. First two bits are the control bits, third the target bit.

Examples

\begin{verbatim}
x <- qstate(nbits=3)
z <- CSWAP(c(1,2,3)) * (H(1) * x)
\end{verbatim}

Description

export a circuit to IBM's qiskit python format. Note that only gates can be exported where the correspondence in qiskit is known and well defined. Qiskit can then be used for IBM's QASM to run on real hardware.

Usage

\begin{verbatim}
export2qiskit(object, varname = "qc", filename = "circuit.py",
append = FALSE, import = FALSE)
\end{verbatim}

Arguments

\begin{itemize}
\item \textbf{object} a qstate object
\item \textbf{varname} character. The name of the circuit variable
\item \textbf{filename} character. The filename of the textfile where to store the circuit
\item \textbf{append} boolean. Whether or not to append to the file. For this the file has to exist.
\item \textbf{import} boolean. Shall numpy and qiskit be loaded explicitly?
\end{itemize}

Details

Export to IBM's Qiskit

Currently the following gates can be exported: H, X, Y, Z, S, Tgate, Rz, Rx, Ry, CNOT, SWAP, CCNOT, CSWAP, measure.

note that only standard gates can be exported, not self defined ones. The function will draw a warning in case a gate cannot be exported and indicate it in the output file.
genComputationalBasis

Value

nothing is returned, but a file is created.

References

https://qiskit.org/documentation/

Examples

```
x <- qstate(2)
x <- H(1) * x
x <- X(2) * x
x <- CNOT(c(1,2)) * x
export2qiskit(measure(x,1)$psi)
cat(readLines("circuit.py"), sep = '\n')
file.remove("circuit.py")
```

---

genComputationalBasis  genComputationalBasis

description

function to generate the basis strings for given number of bits

Usage

genComputationalBasis(nbits, collapse = "")

Arguments

- nbits: integer. The number of qubits
- collapse: character. String to fill in between separate bits

Value

a character vector of length $2^nbits$

Examples

```
genComputationalBasis(4)
genComputationalBasis(2, collapse=">|")
```
**Description**

function to generate the bit representation for a specific basis state

**Usage**

```plaintext
genStateNumber(int, nbits)
```

**Arguments**

- `int` integer number representing the basis state
- `nbits` integer. The number of qubits

**Value**

a integer vector of length `nbits`

**Examples**

```plaintext
genStateNumber(5, 4)
genStateNumber(2, 2)
```

---

**Description**

function to generate the string for a specific basis state

**Usage**

```plaintext
genStateString(int, nbits, collapse = "")
```

**Arguments**

- `int` integer number representing the basis state
- `nbits` integer. The number of qubits
- `collapse` character. String to fill in between separate bits

**Value**

a character
H

The Hadarmard gate

Description

The Hadarmard gate

Usage

H(bit)

Arguments

bit integer. The bit to which to apply the gate

Value

An S4 class 'sqgate' object is returned

Examples

x <- qstate(nbits=2)
z <- H(1) * x
z

hist.measurement

Plot the histogram of a quantum measurement

Description

Plot the histogram of a quantum measurement

Usage

## S3 method for class 'measurement'

hist(x, only.nonzero = TRUE, by.name = only.nonzero, freq = TRUE, ...)

Examples

genStateString(5, 4)
genStateString(2, 2, collapse=">|")
Arguments

- **x**: object as returned by `measure`
- **only.nonzero**: are the states with zero measurements to be plotted?
- **by.name**: shall the xlabel contain the basis names? If `FALSE`, the index number is used.
- **freq**: shall the total counts be plotted? If not, the values are normalised to 1.
- **...**: Generic parameters to pass on to `barplot()`

Value

No return value.

---

**Id**

*The identity gate*

Description

The identity gate

Usage

`Id(bit)`

Arguments

- **bit**: integer. The bit to which to apply the gate

Value

An S4 class `sqgate` object is returned

Examples

```r
x <- qstate(nbits=2)
z <- Id(1) * x
z
```
**is.bitset**

**Description**

checks whether or not a bit is set in target

**Usage**

`is.bitset(x, bit)`

**Arguments**

- **x**: target vector
- **bit**: integer. The bit to check

**Value**

a boolean vector

---

**measure**

**Method measure**

**Description**

performs a measurement on a qstate object.

**Usage**

`measure(e1, bit = NA, repetitions = NA)`

```r
## S4 method for signature 'qstate'
measure(e1, bit = NA, repetitions = 1)
```

**Arguments**

- **e1**: object to measure
- **bit**: bit to project on
- **repetitions**: number of measurements

**Details**

`measure(e1, bit, repetitions)` performs repetitions many projections/measurements of the qubit bit. If bit is not given explicitly, all qubits are projected.
Value

measure(e1, bit, repetitions) returns a list with the measured bit, the number of repetitions, the probability distribution of all states prob and the results vector value. If all bits are measured, the basis is added to the list as basis. The collapsed state is stored as psi if exactly one measurement is performed. In the case of a single qubit measurement value is of length repetitions and contains all the results of this projection. Otherwise value is of length $2^n$bits and it contains the counts how often each state has been obtained.

Examples

```r
## measure the separate bits
x <- H(1) * (H(2) * qstate(nbits=2))
summary(measure(x, bit=1))
hist(measure(x, rep=100))
```

Description

Normalises a complex vector to 1

Usage

```
normalise(x)
```

Arguments

- **x**: complex valued vector

Value

Returns the normalised complex valued vector

Description

phase estimation algorithm

Usage

```
phase_estimation(bitmask, FUN, x, ...)
```
Arguments

bitmask  integer. Vector of qubits for the t qubit wide register needed for the phase estimation

FUN  a function implementing the controlled application of a unitary operator U to the power $2^{(j-1)}$ to the state $x$. It’s first argument must be the control qubit ‘c’, the second the integer ‘j’ and the third the state ‘x’. Additional parameters can be passed via ‘...’.

x  a ‘qstate’ object

...  additional parameter to be passed on to ‘FUN’

Examples

```r
## NOT^k = Id if k even
cnotwrapper <- function(c, j, x, t) {
  if(j == 1) return(CNOT(c(c, t)) * x)
  return(Id(t) * x)
}
x <- X(1) * qstate(3)
## X has eigenvalues lambda=1 and lambda=-1
## thus phases 0 and 1/2
x <- phase_estimation(bitmas=c(2:3), FUN=cnotwrapper, x=x, t=1)
x
```

Description

Plots a circuit corresponding to a qstate object

Usage

```r
## S4 method for signature 'qstate,missing'
plot(x, y, ...)
```

Arguments

x  qstate object

y  not used here

...  additional parameters to be passed on

Value

nothing is returned, but a plot created
Examples
x <- qstate(2)
y <- H(1) * x
z <- CNOT(c(1,2)) * y
plot(z)

qft

Description
performs the quantum Fourier Trafo on the qstate x and the specified list of qubits.

Usage
qft(x, inverse = FALSE, bits)

Arguments
x  qstate
inverse  boolean. If 'TRUE', perform inverse transform
bits  integer. list of qubits to include in the trafo. if missing, bits=c(1:n) is assumed, with n the number of qubits in x.

Details
Quantum Fourier Trafo
The Fourier Trafo is defined as

|j >= - > \sum_{k=0}^{N-1} \frac{1}{\sqrt{N}} \exp(2\pi i j k / N) |k>

the inverse with the oposite sign in the exponential.

Value
a qstate object with the quantum Fourier trafo of input x.

Examples
x <- qstate(3)
y <- qft(x)
z <- qft(y, inverse=TRUE)
The qsimulatR Package

Description
A simulator for a quantum computer

Details
A quantum computer simulator framework. General single qubit gates and general controlled single qubit gates can be easily defined. For convenience, it currently directly provides most common gates (X, Y, Z, H, S, T, Rx, Ry, Rz, CNOT, SWAP, toffoli or CCNOT, CSWAP). 'qsimulatR' supports plotting of circuits and is able to export circuits into IBM's 'Qiskit' python package, which can be run on IBM's real quantum hardware. 'qsimulatR' currently works for up to 24 qubits (a virtual restriction, which can be lifted).

Author(s)
Johann Ostemeyer, Carsten Urbach, <urbach@hiskp.uni-bonn.de>

The qstate class

Description
This class represents a quantum state

Details
The qubits are counted from 1 to nbits starting with the least significant bit.

Slots

nbits The number of qubits
coefs The $2^{\text{nbits}}$ complex valued vector of coefficients
basis String or vector of strings. A single string will be interpreted as the collapse-parameter in genComputationalBasis. A vector of length $2^{\text{nbits}}$ yields the basis directly.
circuit List containing the number of non-quantum bits ncbits and a list of gates gateList applied to the original state. Filled automatically as gates are applied, required for plotting.
Examples

```r
x <- qstate(nbits=2)
x
x <- qstate(nbits=2, coefs=as.complex(sqrt(rep(0.25, 4))), basis=",")
x
x <- qstate(nbits=1, coefs=as.complex(sqrt(rep(0.5, 2))), basis=c("|dead>", "|alive>"))
x
```

---

**Ri**

*The Ri gate*

**Description**

The Ri gate

**Usage**

```r
Ri(bit, i, sign = +1)
```

**Arguments**

- `bit` integer. The bit to which to apply the gate
- `i` integer
- `sign` integer

**Details**

Implements the gate \( \begin{pmatrix} 1 & 0 \\ 0 & \exp(-2\pi i/2^i) \end{pmatrix} \)

If `sign < 0`, the inverse of the exponential is used. This gate is up to global phase identical with the 'Rz' gate with specific values of the angle.

**Value**

An S4 class 'sqgate' object is returned

**Examples**

```r
x <- X(1) * qstate(nbits=2)
z <- Ri(1, i=2) * x
z
```
Rx

The Rx gate

Description
The Rx gate

Usage
Rx(bit, theta = 0)

Arguments
- bit: integer. The bit to which to apply the gate
- theta: numeric. angle

Value
An S4 class 'sqgate' object is returned

Examples
x <- qstate(nbits=2)
z <- Rx(1, pi/4) * x
z

Ry

The Ry gate

Description
The Ry gate

Usage
Ry(bit, theta = 0)

Arguments
- bit: integer. The bit to which to apply the gate
- theta: numeric. angle

Value
An S4 class 'sqgate' object is returned
Examples

```r
x <- qstate(nbits=2)
z <- Ry(1, pi/4) * x
z
```

Rz  

**The Rz gate**

Description

The Rz gate

Usage

```r
Rz(bit, theta = 0)
```

Arguments

- `bit` integer. The bit to which to apply the gate
- `theta` numeric. angle

Value

An S4 class `sqgate` object is returned

Examples

```r
x <- qstate(nbits=2)
z <- Rz(1, pi/4) * x
z
```

S  

**The S gate**

Description

The S gate

Usage

```r
S(bit)
```

Arguments

- `bit` integer. The bit to which to apply the gate
An S4 class 'sqgate' object is returned

Examples

```r
x <- X(1) * qstate(nbits=2)
z <- S(1) * x
z
```

Description

This class represents a generic single qubit gate

Details

The qubits are counted from 1 to nbits starting with the least significant bit.

Slots

- `bit` Integer. The single bit to act on.
- `M` complex valued array. The 2x2 matrix representing the gate
- `type` a character vector representing the type of gate

Examples

```r
x <- qstate(nbits=2)
## application of the X (NOT) gate to bit 1
z <- sqgate(bit=1L, M=matrix(as.complex(c(0,1,1,0)), dim=c(2,2))) * x
z
```
summarize.measurement  
*Summarize a quantum measurement*  

**Description**  
Summarize a quantum measurement  

**Usage**  
```r  
## S3 method for class 'measurement'
summary(object, ...)
```

**Arguments**  
- `object`: as returned by `measure`  
- `...`: Generic parameters to pass on, not used here.  

**Value**  
No return value.

---

**SWAP**  
*The SWAP gate*  

**Description**  
The SWAP gate  

**Usage**  
```r  
SWAP(bits = c(1, 2))
```

**Arguments**  
- `bits`: integer vector of length two, containing the bits to swap.  

**Value**  
An S4 class 'swapgate' object is returned
**swapgate**

*The SWAP gate*

**Description**

This class represents a generic SWAP gate.

**Slots**

- **bits** Integer vector of length 2. The two bits to swap.

**Examples**

```r
x <- H(1) * qstate(nbits=2)
z <- SWAP(c(1,2)) * (H(1) * x)
```

---

**Tgate**

*The Tgate gate*

**Description**

The Tgate gate

**Usage**

```r
Tgate(bit)
```

**Arguments**

- **bit** integer. The bit to which to apply the gate

**Value**

An S4 class 'sqgate' object is returned

**Examples**

```r
x <- X(1) * qstate(nbits=2)
z <- Tgate(1) * x
z
```
Method truth.table

Usage

truth.table(e1, nbits, bits, ...)

Arguments

e1  
gate to measure.

nbits  
number of bits the gate acts on.

bits  
onoptional vector of length nbits containing the qubit order in the gate.

...  
additional parameters to passed be on to 'e1'

Details

calculates the quantum truth table of the gate e1. If a basis state is transformed to a superposition of basis states by the gate, the result is 'NA'.

Value

returns a data.frame containing the truth table. Each row corresponds to one input-output combination. Each column to one specific bit.

Examples

## truth table for a single bit gate
truth.table(X, 1)

## for a 2-bit gate
truth.table(CNOT, 2)

## for a 2-bit gate with reversed controll and target bits
truth.table(CNOT, bits=2:1)

## for a general controlled gate
truth.table(cogate, 2, gate=H(2))

## for an arbitrary circuit (here a swap implementation using only CNOTs)
myswap <- function(bits){ function(x){ CNOT(bits) * (CNOT(rev(bits)) * (CNOT(bits) * x))}}
truth.table(myswap, 2)
The $X$ gate

**Description**

The $X$ gate

**Usage**

$X(\text{bit})$

**Arguments**

bit integer. The bit to which to apply the gate

**Value**

An S4 class 'sqgate' object is returned

**Examples**

```r
x <- qstate(nbits=2)
z <- X(1) * x
z
```

---

The $Y$ gate

**Description**

The $Y$ gate

**Usage**

$Y(\text{bit})$

**Arguments**

bit integer. The bit to which to apply the gate

**Value**

An S4 class 'sqgate' object is returned
Examples

```r
x <- qstate(nbits=2)
z <- Y(1) * x
z
```

---

**Z**  
_The Z gate_

---

**Description**

The Z gate

**Usage**

`Z(bit)`

**Arguments**

- `bit` integer. The bit to which to apply the gate

**Value**

An S4 class 'sqgate' object is returned

**Examples**

```r
x <- X(1) * qstate(nbits=2)
z <- Z(1) * x
z
```
Index

* package
  qsimulatR, 21
  *, ccnotgate, qstate-method, 3
  *, ccqgate, qstate-method, 3
  *, cnotgate, qstate-method, 4
  *, complex, qstate-method, 4
  *, cqgate, qstate-method, 5
  *, cswapgate, qstate-method, 5
  *, matrix, qstate-method, 6
  *, sqgate, qstate-method, 6
  *, swapgate, qstate-method, 7

  CCNOT, 7
  ccnotgate, 8
  ccnotgate-class (ccnotgate), 8
  ccqgate, 8
  ccqgate-class (ccqgate), 8
  CNOT, 9
  cnotgate, 9
  cnotgate-class (cnotgate), 9
  cqft, 10
  cqgate, 11
  cqgate-class (cqgate), 11
  CSWAP, 11
  cswapgate, 12
  cswapgate-class (cswapgate), 12

  export2qiskit, 12

  fredkin (CSWAP), 11

  genComputationalBasis, 13
  genStateNumber, 14
  genStateString, 14

  H, 15
  hist.measurement, 15

  Id, 16
  is.bitset, 17

  measure, 17
  measure, qstate-method (measure), 17

  normalise, 18

  phase_estimation, 18
  plot (plot, qstate, missing-method), 19
  plot, qstate, missing-method, 19

  qft, 20
  qsimulatR, 21
  qstate, 21
  qstate-class (qstate), 21

  Ri, 22
  Rx, 23
  Ry, 23
  Rz, 24

  S, 24
  sqgate, 25
  sqgate-class (sqgate), 25
  summary.measurement, 26
  SWAP, 26
  swapgate, 27
  swapgate-class (swapgate), 27

  Tgate, 27
  toffoli (CCNOT), 7
  truth.table, 28

  X, 29
  Y, 29
  Z, 30