

Package: Unitary (via r-universe)

July 6, 2026

Title Quantum Simulator

Version 0.3.11

Maintainer Zengchao Xu <zengc.xu@aliyun.com>

Description Provides a comprehensive toolkit for quantum computing simulation and visualization within the R environment. The package enables users to initialize qubit states, construct custom quantum gates with both unitary transformation and visual parameters, and build full quantum circuits by sequentially adding gates. It includes predefined common gates (e.g., Hadamard, Pauli-X/Y/Z, Control-NOT, Control-Z) and supports direct plotting of circuits and individual gates for intuitive analysis.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 8.0.0

Imports Formula, grid, gtable, latex2exp, data.table

Suggests testthat (>= 3.0.0)

Depends R (>= 4.1.0)

NeedsCompilation no

Author Zengchao Xu [aut, cre, cph]

Repository <https://cran.r-universe.dev>

Date/Publication 2026-07-06 14:10:34 UTC

RemoteUrl <https://github.com/cran/Unitary>

RemoteRef HEAD

RemoteSha 8920049d7ace59aed4fee556c95d721c9b04eb48

Contents

add_gate	2
build_circuit_layout	3

gate-builders	3
plot.qcircuit	5
print.qcircuit	6
qcircuit	7
qgates	8
set_qubit	11
set_qubits	12
validate_qubits	13

Index	14
--------------	-----------

add_gate	<i>Add Gate to Quantum Circuit</i>
----------	------------------------------------

Description

It accepts a gate name, a gate object constructed by `set_gate()`, or a reusable gate adder function.

Usage

```
add_gate(
    circuit,
    gate,
    targets = NULL,
    controls = integer(0),
    step = NULL,
    ...
)
```

Arguments

circuit	A qcircuit object.
gate	Gate adder or gate label.
targets	Target qubit indices.
controls	Control qubit indices.
step	Optional insertion step index.
...	Parameter passed to <code>set_gate</code> . When gate is a character of length one, <code>set_gate(target_label=gate, ...)</code> is called to build a gate adder.

Value

A modified qcircuit object.

See Also

[set_gate](#)

build_circuit_layout *Circuit Layout*

Description

Build circuit layout of qubit object.

Usage

```
build_circuit_layout(qubits, steps)
```

Arguments

qubits	A qubit object.
steps	Steps.

Value

Layout object.

gate-builders *Common Single-qubit Gate Constructors and Wrapped Gates*

Description

- `set_gate()` builds a reusable gate descriptor that contains drawing parameters and prebuilt grobs. The returned object can be inspected or applied to a `qcircuit` via the returned `adder` function.
- `gate_*()` create reusable gate adders for common gates and forward their arguments to `set_gate()`.

Usage

```
set_gate(
  target_label = character(),
  target_label_col = "black",
  target_label_gpar = grid::gpar(col = target_label_col, cex = 2),
  target_shape = "roundrect",
  target_col = "black",
  target_fill = "white",
  target_gpar = grid::gpar(col = target_col, fill = target_fill, lwd = 2),
  control_label = character(),
  control_label_col = "black",
  control_label_gpar = grid::gpar(col = control_label_col, cex = 2),
  control_shape = "dot",
```

```

control_col = "black",
control_fill = "black",
control_gpar = grid::gpar(col = control_col, fill = control_fill, lwd = 2),
gate_label = character(),
gate_label_col = "black",
gate_label_gpar = grid::gpar(col = gate_label_col, cex = 1.5),
targets = NULL,
controls = integer(0),
wire_lwd = NA_real_,
wire_gpar = grid::gpar(col = "black", lwd = wire_lwd),
unitary = list(f = function() {
  NULL
}, U = matrix(complex(), nrow = 0, ncol =
  0)),
  ...
)

```

Arguments

target_label	Label displayed inside the gate box.
target_label_col	Text color.
target_label_gpar	grid::gpar used for label text appearance.
target_shape	Target foreground style: one of "roundrect", "rect", "circle", "dot", or "x" (cross).
target_col	Border color.
target_fill	Background fill color.
target_gpar	grid::gpar used to draw target shapes (if gate_gpar is provided it will be used as default).
control_label	Label displayed inside the gate box.
control_label_col	Text color.
control_label_gpar	grid::gpar used for label text appearance.
control_shape	Control decoration style: one of "circle", "rect", "roundrect", "dot", or "x" (cross).
control_col	Border color.
control_fill	Fill color.
control_gpar	grid::gpar used to draw control shapes.
gate_label	Label displayed above the gate box.
gate_label_col	Text color.
gate_label_gpar	grid::gpar used for label text appearance.
targets	Optional preset target indices; if provided the returned adder will have these targets attached as attributes.

controls	Optional preset control indices; if provided the returned adder will have these controls attached as attributes.
wire_lwd	Line width for wires connecting controls and targets.
wire_gpar	grid::gpar used to draw wire between target and control dots.
unitary	A list describing the gate's unitary (should contain f and U).
...	Additional parameters passed to target or control graphical objects, including <ul style="list-style-type: none"> • dot: pch; see points. • dot, x and oplus: size. • circle: radius. • rect or roundrect: height, width.

Value

A qgate_adder function with attribute gate (the descriptor). The adder has signature `adder(circuit, targets, controls, step = NULL)`.

plot.qcircuit	<i>Plot Quantum Circuit</i>
---------------	-----------------------------

Description

Plot a quantum circuit.

Usage

```
## S3 method for class 'qcircuit'
plot(x, ...)
```

Arguments

x	A qcircuit object.
...	Further arguments.

Value

A gtable object invisibly.

Examples

```
c <- qcircuit(3) |>
  add_gate("H", targets = 1) |>
  add_gate("CNOT", targets = 2, controls = 1) |>
  add_gate("SWAP", targets = c(2, 3))
plot(c)

# example with custom gate gpar
```

```
gp <- grid::gpar(fill = "pink", col = "purple")
h2 <- set_gate("H", targets = 1, target_gpar = gp)
c <- qcircuit(2)
c <- h2(c)
plot(c)
```

print.qcircuit

Print Quantum Circuit Summary and Layout Information

Description

Print Quantum Circuit Summary and Layout Information

Usage

```
## S3 method for class 'qcircuit'
print(x, layout = FALSE, ...)
```

Arguments

x	A qcircuit object.
layout	If TRUE, prints detailed gtable structure information of circuit layout.
...	Unused arguments.

Value

The original circuit object invisibly.

Examples

```
c <- qcircuit(2)
c <- add_gate(c, "H", targets = 1)
print(c, layout = TRUE)
```

 qcircuit

 Create Quantum Circuit Container

Description

A circuit stores a qubit set and an ordered sequence of gate operations. The internal layout is implemented as a `gtable` for later rendering.

Usage

```
qcircuit(n = NULL, qubits = NULL, ...)

is_qcircuit(x)
```

Arguments

<code>n</code>	Number of qubits in the circuit.
<code>qubits</code>	A qubits object created by set_qubits .
<code>...</code>	Optional arguments for set_qubits .
<code>x</code>	A circuit object.

Value

- `qcircuit` returns a `qcircuit` object.
- `is_qcircuit` returns a `bool` value.

Examples

```
c <- qcircuit(4) |>
  gate_cnot(targets = 4, controls = 1) |>
  gate_cnot(targets = 4, controls = 2) |>
  gate_cnot(targets = 4, controls = 3) |>
  add_gate("X", targets = 4, controls = 1) |>
  add_gate("X", targets = 3, target_gpar = grid::gpar(fill="gold2"))
plot(c)

c <- gate_swap(c, targets = 3:4,
  gate_label = "Phi",
  gate_label_gpar=grid::gpar(col="red", cex=3),
  wire_gpar = grid::gpar(col="blue", lwd=3))
plot(c)

c <- gate_toffoli(c, 3, c(2, 4))
plot(c)

c <- gate_cz(c, 4, 2,
  target_label = "e^{-i \\pi Z}",
  control_gpar = grid::gpar(fill="blue", col="blue"),
```

```

        wire_gpar = grid::gpar(col="blue", lwd=3))
plot(c)

gate_eg <- set_gate(
  "e^{-i \\pi Z}",
  target_gpar = grid::gpar(fill = "lightgreen", col = "darkgreen"),
  wire_gpar = grid::gpar(lwd=0)
)
c <- gate_eg(c, targets = 2:3)
plot(c)

c <- add_gate(c, "\\int f(x) dx", targets=3, step=9,
  target_shape = "circle",
  target_gpar = grid::gpar(fill="lightblue", col="darkblue"),
  target_label_gpar = grid::gpar(col="red", cex=1.0))
plot(c)

```

qgates

Common Single-qubit Gate Constructors and Wrapped Gates

Description

Common Single-qubit Gate Constructors and Wrapped Gates

Usage

```

gate_h(
  circuit,
  targets,
  step = NULL,
  unitary = list(f = function() {
    NULL
  }, U = matrix(complex(), nrow = 2, ncol =
    2)),
  ...
)

gate_x(
  circuit,
  targets,
  step = NULL,
  unitary = list(f = function() {
    NULL
  }, U = matrix(complex(), nrow = 2, ncol =
    2)),
  ...
)

```

```
gate_y(  
  circuit,  
  targets,  
  step = NULL,  
  unitary = list(f = function() {  
    NULL  
  }, U = matrix(complex(), nrow = 2, ncol =  
    2)),  
  ...  
)
```

```
gate_z(  
  circuit,  
  targets,  
  step = NULL,  
  unitary = list(f = function() {  
    NULL  
  }, U = matrix(complex(), nrow = 2, ncol =  
    2)),  
  ...  
)
```

```
gate_s(  
  circuit,  
  targets,  
  step = NULL,  
  unitary = list(f = function() {  
    NULL  
  }, U = matrix(complex(), nrow = 2, ncol =  
    2)),  
  ...  
)
```

```
gate_t(  
  circuit,  
  targets,  
  step = NULL,  
  unitary = list(f = function() {  
    NULL  
  }, U = matrix(complex(), nrow = 2, ncol =  
    2)),  
  ...  
)
```

```
gate_i(  
  circuit,  
  targets,
```

```
    step = NULL,
    unitary = list(f = function() {
      NULL
    }, U = matrix(complex(), nrow = 2, ncol =
      2)),
    ...
  )

gate_swap(
  circuit,
  targets,
  step = NULL,
  unitary = list(f = function() {
    NULL
  }, U = matrix(complex(), nrow = 4, ncol =
    4)),
  ...
)

gate_cnot(
  circuit,
  targets,
  controls,
  step = NULL,
  unitary = list(f = function() {
    NULL
  }, U = matrix(complex(), nrow = 4, ncol =
    4)),
  ...
)

gate_cz(
  circuit,
  targets,
  controls,
  step = NULL,
  unitary = list(f = function() {
    NULL
  }, U = matrix(complex(), nrow = 4, ncol =
    4)),
  ...
)

gate_toffoli(
  circuit,
  targets,
  controls,
  step = NULL,
```

```

    unitary = list(f = function() {
      NULL
    }, U = matrix(complex(), nrow = 8, ncol =
      8)),
    ...
  )

```

Arguments

circuit	A qcircuit object.
targets	Target qubit indices.
step	Optional insertion step index.
unitary	See set_gate .
...	Additional parameters passed to set_gate .
controls	Control qubit indices (for multi-qubit gates).

Details

SWAP gate operates on two target qubits and no explicit controls

gate_cnot: CX gate

gate_cz: CZ gate

Toffoli: CCNOT or CCX gate

Value

A modified qcircuit object.

set_qubit	<i>Create Single Qubit Description</i>
-----------	----------------------------------------

Description

This helper is useful when building qubit sets programmatically.

Usage

```

set_qubit(
  qubits,
  index = NULL,
  name = NULL,
  label = NULL,
  value = NULL,
  label_col = "black",
  label_size = 10,
  wire_col = "grey40",

```

```

    label_gpar = grid::gpar(col = label_col, fontsize = label_size),
    wire_gpar = grid::gpar(col = wire_col, lwd = 1)
  )

```

Arguments

qubits	A qubits object to update.
index	Qubit index.
name	Qubit name.
label	Text label for the qubit.
value	Optional value to set for the selected qubit (complex or length 1 numeric).
label_col	Label text color (convenience).
label_size	Label font size (convenience).
wire_col	Wire color (convenience).
label_gpar	Optional <code>grid::gpar</code> to control label appearance (overrides label_col/label_size).
wire_gpar	Optional <code>grid::gpar</code> to control wire appearance (overrides wire_col).

Value

A qubit object.

Examples

```

qubits <- set_qubits(2)
qubits <- set_qubit(qubits, index = 1, label = "|1>",
  label_gpar = grid::gpar(col = "red"))

```

set_qubits

Create Qubit Set

Description

set_qubits() defines the qubit names, labels and visual styles for the circuit wires.

Usage

```

set_qubits(
  n,
  labels = NULL,
  names = NULL,
  values = complex(),
  label_col = "black",
  label_size = 1.5,
  wire_col = "grey40",
  label_gpar = grid::gpar(col = label_col, cex = label_size),
  wire_gpar = grid::gpar(col = wire_col, lwd = 2)
)

```

Arguments

n	Number of qubits.
labels	Optional labels for each qubit.
names	Optional names for each qubit.
values	States of qubits.
label_col	Label text color (convenience).
label_size	Label font size (convenience).
wire_col	Wire color (convenience).
label_gpar	Optional <code>grid::gpar</code> to control label appearance (overrides label_col/label_size).
wire_gpar	Optional <code>grid::gpar</code> to control wire appearance (overrides wire_col).

Value

A qubits object.

Examples

```
# create qubits with default appearance
q <- set_qubits(2, labels = c("0", "1"))

# create qubits with explicit gpar for labels and wire
q2 <- set_qubits(3, labels = c("0", "1", "0"),
  label_gpar = grid::gpar(col = "blue", fontsize = 12),
  wire_gpar = grid::gpar(col = "grey60", lwd = 2))
```

validate_qubits	<i>Validate Qubits</i>
-----------------	------------------------

Description

Validate qubits.

Usage

```
validate_qubits(qubits)
```

Arguments

qubits	A qubits object.
--------	------------------

Value

qubits object.

Index

`add_gate`, [2](#)

`build_circuit_layout`, [3](#)

`gate-builders`, [3](#)

`gate_cnot` (`qgates`), [8](#)

`gate_cz` (`qgates`), [8](#)

`gate_h` (`qgates`), [8](#)

`gate_i` (`qgates`), [8](#)

`gate_s` (`qgates`), [8](#)

`gate_swap` (`qgates`), [8](#)

`gate_t` (`qgates`), [8](#)

`gate_toffoli` (`qgates`), [8](#)

`gate_x` (`qgates`), [8](#)

`gate_y` (`qgates`), [8](#)

`gate_z` (`qgates`), [8](#)

`grid::gpar`, [4](#), [5](#), [12](#), [13](#)

`is_qcircuit` (`qcircuit`), [7](#)

`plot.qcircuit`, [5](#)

`points`, [5](#)

`print.qcircuit`, [6](#)

`qcircuit`, [7](#)

`qgates`, [8](#)

`quantum-circuit` (`qcircuit`), [7](#)

`set_gate`, [2](#), [11](#)

`set_gate` (`gate-builders`), [3](#)

`set_qubit`, [11](#)

`set_qubits`, [7](#), [12](#)

`validate_qubits`, [13](#)