

# Package ‘KeyboardSimulator’

January 13, 2021

**Type** Package

**Title** Keyboard and Mouse Input Simulation for Windows OS

**Version** 2.6.0

**Date** 2021-01-09

**Description** Control your keyboard and mouse with R code by simulating key presses and mouse clicks. The input simulation is implemented with the Windows API.

**License** GPL (>= 2) | file LICENSE

**Imports** Rcpp (>= 1.0.0)

**LinkingTo** Rcpp

**RoxygenNote** 7.1.1

**Encoding** UTF-8

**Depends** R (>= 2.10)

**OS\_type** windows

**URL** <https://github.com/ChiHangChen/KeyboardSimulator>

**BugReports** <https://github.com/ChiHangChen/KeyboardSimulator/issues>

**NeedsCompilation** yes

**Author** Jim Chen [aut, cre],  
Jeff Keller [aut, ctb],  
Garry Hopwood [ctb],  
Chieh Hsu [ctb]

**Maintainer** Jim Chen <jim71183@gmail.com>

**Repository** CRAN

**Date/Publication** 2021-01-13 13:30:08 UTC

## R topics documented:

keybd.press . . . . .	2
keybd.release . . . . .	3
keybd.type_string . . . . .	3

KeyboardSimulator . . . . .	4
keyboard_value . . . . .	4
mouse.click . . . . .	9
mouse.get_cursor . . . . .	9
mouse.move . . . . .	10
mouse.release . . . . .	11

<b>Index</b>	<b>12</b>
--------------	-----------

---

keybd.press	<i>Simulate Key Press</i>
-------------	---------------------------

---

### Description

Simulate keyboard key presses. Multiple keys can be pressed simultaneously by using + as separator (see Examples). See [keyboard\\_value](#) for supported keys.

### Usage

```
keybd.press(button, hold = FALSE)
```

### Arguments

button	character. The key press to simulate (not case sensitive).
hold	logical. Whether the key should be held down. If TRUE, the key can be released by pressing the physical key on the keyboard or by using the <a href="#">keybd.release</a> function.

### See Also

[keybd.release](#)

### Examples

```
## Not run:

# press one key
keybd.press('a')

# press multiple keys
keybd.press('Alt+F4')

# press multiple keys using hold
keybd.press('Alt', hold = TRUE)
keybd.press('F4')
keybd.release('Alt')

## End(Not run)
```

---

keybd.release	<i>Simulate Key Release</i>
---------------	-----------------------------

---

**Description**

Simulate the release of keyboard keys held by [keybd.press](#). Multiple keys can be released simultaneously by using a + separator (see Examples). See [keyboard\\_value](#) for supported keys.

**Usage**

```
keybd.release(button)
```

**Arguments**

button            character. The key release to simulate (not case sensitive).

**See Also**

[keybd.press](#)

**Examples**

```
## Not run:  
  
# Move to the third working window  
keybd.press('Alt', hold = TRUE)  
keybd.press('Tab')  
Sys.sleep(0.1)  
keybd.press('Tab')  
keybd.release('Alt')  
  
## End(Not run)
```

---

keybd.type_string	<i>Type a raw string</i>
-------------------	--------------------------

---

**Description**

Type a raw string base on a given string.

**Usage**

```
keybd.type_string(string)
```

**Arguments**

string            character. The string expected to output (case sensitive).

**Examples**

```
## Not run:

# Type 'Hello world!'
keybd.type_string("Hello world!")

## End(Not run)
```

---

KeyboardSimulator	<i>Keyboard and Mouse Input Simulation for Windows OS</i>
-------------------	---

---

**Description**

Control your keyboard and mouse with R code by simulating key presses and mouse clicks. The input simulation is implemented with the Windows API.

**Author(s)**

Jim Chen, Jeff Keller

---

keyboard_value	<i>keyboard_value</i>
----------------	-----------------------

---

**Description**

List of supported keyboard keys, along with their virtual-key and hardware scan codes. A field indicating whether the key is an "extended key" with a 0xE0 prefix byte is also included to differentiate between duplicate keys on the keyboard and num pad. For example, while the 1 key usually behaves the same as the 1 key on the num pad, some applications see these as two distinct keys.

**Usage**

```
keyboard_value
```

**Format**

An object of class `data.frame` with 137 rows and 5 columns.

**Details**

Supported keys:

- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
- u
- v
- w
- x
- y
- z
- A
- B
- C
- D
- E
- F
- G
- H
- I

- J
- K
- L
- M
- N
- O
- P
- Q
- R
- S
- T
- U
- V
- W
- X
- Y
- Z
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- num0
- num1
- num2
- num3
- num4
- num5
- num6
- num7
- num8
- num9

- f1
- f2
- f3
- f4
- f5
- f6
- f7
- f8
- f9
- f10
- f11
- f12
- backspace
- tab
- enter
- shift
- ctrl
- alt
- capslock
- esc
- pageup
- pagedown
- end
- home
- left
- up
- right
- down
- insert
- space
- del
- numlock
- win
- ‘
- ~
- !
- @

- #
- \$
- %
- ^
- &
- \*
- (
- )
- -
- \_
- +
- =
- [
- {
- ]
- }
- \
- |
- ;
- :
- '
- "
- ,
- <
- .
- >
- /
- ?

**References**

Hardware Scan Codes, Virtual-Key Codes



---

`mouse.click`*Simulate Mouse Clicks*

---

**Description**

Simulate left, right, and middle mouse clicks.

**Usage**

```
mouse.click(button = "left", hold = FALSE)
```

**Arguments**

button	character. Allowed values are "left", "right", and "middle".
hold	logical. Whether the button should be held down.

**See Also**

[mouse.release](#)

**Examples**

```
## Not run:  
  
# left mouse click  
mouse.click(button = "left")  
  
# left mouse click and hold  
mouse.click(button = "left", hold = TRUE)  
  
## End(Not run)
```

---

`mouse.get_cursor`*Get Current Cursor Coordinate*

---

**Description**

Get current cursor coordinate of screen .

**Usage**

```
mouse.get_cursor()
```

**Examples**

```
## Not run:  
  
mouse.get_cursor()  
  
## End(Not run)
```

---

mouse.move

*Move Cursor to Specific Location*

---

**Description**

Move cursor to specific coordinate of screen .

**Usage**

```
mouse.move(x, y, duration = NA, step_ratio = 0.01)
```

**Arguments**

x	numeric. X-axis of screen.
y	numeric. Y-axis of screen.
duration	numeric. Cursor movement time in seconds, there might be some delay on different computer.
step_ratio	numeric. Ratio of total distance in each step, only available when duration is not NA.

**Examples**

```
## Not run:  
  
# Move cursor to middle of screen in 1080FHD monitor  
mouse.move(x=960,y=540)  
  
# Move cursor to middle of screen in 1080FHD monitor within 3 seconds  
mouse.move(x=960,y=540,duration=3)  
  
## End(Not run)
```

---

mouse.release	<i>Simulate Mouse Click Release</i>
---------------	-------------------------------------

---

**Description**

Simulate the release of mouse button held by [mouse.click](#).

**Usage**

```
mouse.release(button = "left")
```

**Arguments**

button            character. Allowed values are "left", "right", and "middle".

**See Also**

[mouse.click](#)

**Examples**

```
## Not run:  
  
# right mouse click and hold  
mouse.click(button = "right", hold = TRUE)  
  
# release right click  
mouse.release(button = "right")  
  
## End(Not run)
```

# Index

## \* **data**

    keyboard\_value, 4

keybd.press, 2, 3

keybd.release, 2, 3

keybd.type\_string, 3

keyboard\_value, 2, 3, 4

KeyboardSimulator, 4

mouse.click, 9, 11

mouse.get\_cursor, 9

mouse.move, 10

mouse.release, 9, 11