

# Package: aperol (via r-universe)

February 27, 2025

**Title** Generates Tippy or Drunk Praise

**Version** 0.2.0

**Description** This is a joke package, which generates praise using the  
praise package, then garbles it, as if being delivered by  
someone tipsy or drunk.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2.9000

**URL** <https://github.com/EllaKaye/aperol>,  
<https://ellakaye.github.io/aperol/>

**BugReports** <https://github.com/EllaKaye/aperol/issues>

**Imports** praise (>= 1.0.0)

**Remotes** rladies/praise

**Suggests** devtools, testthat (>= 3.0.0), withr

**Config/testthat/edition** 3

**Repository** <https://ellakaye.r-universe.dev>

**RemoteUrl** <https://github.com/EllaKaye/aperol>

**RemoteRef** HEAD

**RemoteSha** dcd46b39bf93eba9a862674eb43c767792cd1604

## Contents

drunk . . . . .	2
increment_spritz_count . . . . .	2
spritz . . . . .	3
tippy . . . . .	4

<b>Index</b>	<b>5</b>
--------------	----------

---

drunk
*Drunk Praise***Description**

Generates praise with `praise::praise()`, then repeats some words, and permutes them all.

**Usage**

```
drunk(
  repeat_words = 1,
  repeat_times = 2,
  template = "You are ${adverb} ${adjective}!"
)
```

**Arguments**

<code>repeat_words</code>	int scalar, the number of words that get repeated
<code>repeat_times</code>	an integer-valued (non-negative) vector. If <code>length(repeat_times)</code> is <code>repeat_words</code> , gives the number of times each repeated word appears. If length 1, each repeated word appears that number of times. Otherwise, throws an error (see <code>rep()</code> ).
<code>template</code>	character scalar, template for the (non-swapped) praise

**Examples**

```
drunk()
drunk(2, 2:3)
drunk(2, 2, "You are ${creating} a ${adverb} ${adjective} ${rpackage}")
```

---

increment\_spritz\_count
*Increment the spritz count***Description**

Adds a row to the spritz count variable in the the environment

**Usage**

```
increment_spritz_count(func)
```

**Arguments**

<code>func</code>	Character. Function to be evaluated
-------------------	-------------------------------------

**Value**

Invisible. Data frame for spritz\_count

**Examples**

```
increment_spritz_count(func = "1+1")
```

---

spritz

*Spritz*

---

**Description**

A thin wrapper around a function (e.g devtools::check()) that provides affirmation after X calls in Y minutes. Praise gets drunker as the number of actual runs exceeds the number of runs in the time frame.

**Usage**

```
spritz(
  func = "devtools::check()",
  runs = 3,
  minutes = 10,
  template = "You are ${adverb} ${adjective}!"
)
```

**Arguments**

func	Character. A function you expect to run multiple times
runs	Numeric. Number of runs in time frame before praise
minutes	Numeric. Number of minutes in time frame
template	Character. Praise template

**Value**

Returns the output of func

**Examples**

```
## Not run:

# default is to run devtools::check()
spritz()
# with a different function and praise template
spritz("devtools::document()",
  template = "You are ${creating} a ${adverb} ${adjective} ${rpackage}")
# change rate depending on speed of devtools::check()
```

```
spritz(runs = 4, minutes = 15)

## custom function
sleep_add <- function() {
  Sys.sleep(2)
  out <- 1 + 1
  return(out)
}

spritz(func = "sleep_add()")

## End(Not run)
```

---

tipsy

*Tipsy Praise*

---

## Description

Generates praise with `praise::praise()`, then swaps some words.

## Usage

```
tipsy(swaps = 1, template = "You are ${adverb} ${adjective}!")
```

## Arguments

swaps	int scalar, number of word swaps to make
template	character scalar, template for the (non-swapped) praise

## Examples

```
tipsy()
tipsy(2, "You are ${creating} a ${adverb} ${adjective} ${rpackage}")
```

# Index

`drunk`, [2](#)

`increment_spritz_count`, [2](#)

`praise::praise()`, [2](#), [4](#)

`rep()`, [2](#)

`spritz`, [3](#)

`tipsy`, [4](#)