Package 'cowsay'

July 22, 2025

Title Messages, Warnings, Strings with Ascii Animals

Description Allows printing of character strings as messages/warnings/etc. with ASCII animals, including cats, cows, frogs, chickens, ghosts, and more.

Version 1.2.0

License MIT + file LICENSE

URL https://github.com/sckott/cowsay, https://sckott.github.io/cowsay/

BugReports https://github.com/sckott/cowsay/issues

Encoding UTF-8

Language en-US

VignetteBuilder knitr

Imports crayon, rlang

Suggests fortunes, rmsfact, jsonlite, knitr, rmarkdown, testthat

RoxygenNote 7.3.2

NeedsCompilation no

Author Scott Chamberlain [aut, cre],
Amanda Dobbyn [aut],
Tyler Rinker [ctb],
Thomas Leeper [ctb],
Noam Ross [ctb],
Rich FitzJohn [ctb],
Carson Sievert [ctb],
Kiyoko Gotanda [ctb],
Andy Teucher [ctb],
Karl Broman [ctb],
Franz-Sebastian Krah [ctb],
Lucy D'Agostino McGowan [ctb],
Guangchuang Yu [ctb],
Philipp Boersch-Supan [ctb],
Andreas Brandmaier [ctb],
Marion Louveaux [ctb],
David Schoch [ctb]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2025-03-29 00:40:02 UTC

Contents

`animals` *Animals*

Description

Named vector of animals

Details

`animals` is a named character vector of animals, with each element a character string of variable length specifying an ASCII animal. Note that some have unicode characters that won't play well on some operating systems.

Examples

```
animals
cat(animal_fetch('cow'))
cat(animal_fetch('chicken'))
cat(animal_fetch('chuck'))
cat(animal_fetch('clippy'))
cat(animal_fetch('poop'))
cat(animal_fetch('bigcat'))
```

animal_fetch *Fetch an animal*

Description

Fetch an animal

Usage

```
animal_fetch(by)
```

Arguments

by	(character) which animal to get

Value

single string

Examples

```
animal_fetch("longtailcat")
cat(animal_fetch("longtailcat"))
animal_fetch("poop")
cat(animal_fetch("poop"))
```

bubble_say *Thought/speech bubble/balloon*

Description

Thought/speech bubble/balloon

Usage

```
bubble_say(x, width = 60)

bubble_think(x, width = 60)
```

Arguments

x	(character) a character vector
width	(integer/numeric) width of each line. default: 60

Details

bubble_say gives the traditional bubble that you get when you run cowsay on the command line, with carrots or slashes for the sides, while bubble_think gives a slightly different bubble with parens for the sides

Value

character vector of length greater than the input x

Note

modified from https://github.com/schochastics/startifyR

See Also

Other bubble: bubble_tail()

Examples

```
library(fortunes)
quote <- as.character(fortune())
bubble_say(x = quote)

cat(bubble_say(paste(quote, collapse = " ")), sep = "\n")
ch <- animal_fetch('chicken')
z <- paste(c(bubble_say(quote), bubble_tail(ch, "\\"), ch), collapse = "\n")
cat(z)


text_color <- sample(grDevices::colors(), 1)
text_style <- crayon::make_style(text_color)
text_style(bubble_say(quote))
```

---

bubble_tail *Make the tail part of a thought bubble*

Description

Make the tail part of a thought bubble

Usage

```
bubble_tail(animal, thought_sym = "o")

bubble_tail2(max_char_length, thought_sym = "o")
```

Arguments

animal	(character) a string
thought_sym	(character) scalar character to use for the speech bubble tail (see https://en.wikipedia.org/wiki/Speech_balloon). default: "o"
max_char_length	(numeric) length of the maximum line. this is used to determine how much whitespace padding to add to the left of thought_sym

Details

bubble_tail uses the animal as input so that the tail is put close to the top of the animal, whereas bubble_tail2 just puts the tail about a 1/3 of the way from the left most character given the max character length

See Also

Other bubble: bubble_say()

Examples

```
bubble_tail(animal_fetch('chicken'))
cat(bubble_tail(animal_fetch('chicken')), sep = "\n")
cat(bubble_tail(animal_fetch('chicken')), sep = "\n")
cat(bubble_tail(animal_fetch('chicken'), "%"), sep = "\n")

bubble_tail2(59)
cat(bubble_tail2(59), sep = "\n")
cat(bubble_tail2(11), sep = "\n")
cat(bubble_tail2(11, "%"), sep = "\n")
```

---

endless_horse *Endless horse*

Description

Each time you press enter, the horse keeps going...and going...

Usage

```
endless_horse(
  what = "Hello world!",
  endless = TRUE,
  wait = 0.5,
  what_color = NULL,
  horse_color = NULL
)
```

Arguments

what	(character) What do you want to say? See details.
endless	(logical) Should horse be endless, you better say yes. Default: TRUE
wait	How long to wait between leg segments (time grows geometrically after the first iteration in order to keep the horse on screen for a while, but it **will** keep going forever. Or until you hit escape/Ctrl-C depending on your platform).
what_color	(character or crayon function) A crayon-suported text color or crayon style function to color what. You might try colors() or ?rgb for ideas.
horse_color	(character or crayon function) A crayon-suported text color or crayon style function to color your steed.

Examples

```
endless_horse(endless = FALSE)

endless_horse()
endless_horse(what_color = crayon::bgBlue)
```

---

say *Sling messages and warnings with flair*

Description

Sling messages and warnings with flair

Usage

```
say(
  what = "Hello world!",
  by = "cow",
  type = NULL,
  what_color = NULL,
  by_color = what_color,
  length = 18,
  fortune = NULL,
  width = 60,
  ...
)

think(
  what = "Hello world!",
  by = "cow",
  type = NULL,
  what_color = NULL,
  by_color = what_color,
```

```
  length = 18,
  fortune = NULL,
  width = 60,
  ...
)
```

Arguments

what	(character) What do you want to say? See Details.
by	(character) Who should say or think it? One of: alligator, ant, anxiouscat, bat, bat2, beavis, behindcat, bigcat, blowfish, buffalo, cat, chicken, chuck, clippy, cow, cow_borg, cow_dead, cow_greedy, cow_sleepy, cow_tired, cow_wired, cow_young, daemon, dragon, duck, duckling, egret, endlesshorse, facecat, fish, frog, ghost, goldfish, grumpycat, hypnotoad, longcat, longtailcat, monkey, mushroom, owl, pig, poop, pumpkin, rabbit, rms, shark, shortcat, signbunny, smallcat, snowman, spider, squirrel, squirrel2, stegosaurus, stretchycat, trilobite, turkey, whale, wolf, yoda. Alternatively, use 'random' to have your message spoken by a random character. We use `rlang::arg_match()` internally, which does not support partial matching, so you'll get an informative error upon a partial match.
type	(character) One of message (default), warning, print (default in non-interactive mode), or string (returns string). If run in non-interactive mode default type is print, so that output goes to stdout rather than stderr, where messages and warnings go.
what_color	(character or crayon function) One or more `crayon`-suported text color(s) or `crayon style function` to color `what`. You might try `colors()` or `?rgb` for ideas. Use "rainbow" for c("red", "orange", "yellow", "green", "blue", "purple").
by_color	(character or crayon function) One or more `crayon`-suported text color(s) or `crayon style function` to color `who`. Use "rainbow" for `c("red", "orange", "yellow", "green", "blue", "purple")`. By default is set to be whatever color `what_color` is so you can have the same color for both with less typing.
length	(integer) Length of longcat. Ignored if other animals used.
fortune	An integer (or number that can be coerced to integer) specifying a fortune from the `fortunes` package - OR a string which is used as a pattern passed to `grep()` (and a random one is selected upton multiple matches). Passed on to the `which` parameter of `fortunes::fortune`
width	(integer/numeric) width of each line. default: 60
...	Further args passed on to `fortunes::fortune()`

what

You can put in any phrase you like to the `what` parameter, OR you can type in one of a few special phrases that do particular things. They are:

- "catfact": A random cat fact from https://catfact.ninja
- "fortune": A random quote from an R coder, from fortunes library

- "time": Print the current time
- "rms": Prints a random 'fact' about Richard Stallman from the `rmsfact::rmsfact()` package. Best paired with `by = "rms"`.

by

Note that if you choose `by='hypnotoad'` the quote is forced to be, as you could imagine, 'All Glory to the HYPNO TOAD!'. For reference see http://knowyourmeme.com/memes/hypnotoad

signbunny: It's not for sure known who invented signbunny, but this article http://www.vox.com/2014/9/18/6331753/sign-bunny-meme-explained thinks they found the first use in this tweet: https://twitter.com/wei_bluebear/status/32910164578077

trilobite: from http://www.retrojunkie.com/asciiart/animals/dinos.htm (site down though)

Note to Windows users: there are some animals (shortcat, longcat, fish, signbunny, stretchycat, anxiouscat, longtailcat, grumpycat, mushroom) that are not available because they use non-ASCII characters that don't display properly in R on Windows.

Examples

```
say()
say("what")
say("time")

say("who you callin chicken", "chicken")
say("ain't that some shit", "poop")
say("icanhazpdf?", "cat")
say("boo!", "pumpkin")
say("hot diggity", "frog")

# Vary type of output, default calls message()
say("hell no!")
say("hell no!", type = "warning")
say("hell no!", type = "string")

# The hypnotoad
say(by = "hypnotoad")

# Trilobite
say(by = "trilobite")

# Shark
say("Q: What do you call a solitary shark\nA: A lone shark", by = "shark")

# Buffalo
say("Q: What do you call a single buffalo?\nA: A buffalonely", by = "buffalo")


# Using fortunes
library(fortunes)
say(what = "fortune")
## you don't have to pass anything to the `what` parameter if `fortune` is
## not null
```

```
say("fortune", "spider")
say("fortune", "facecat")
say("fortune", "behindcat")
say("fortune", "smallcat")
say("fortune", "monkey")
say("fortune", "egret")
say(fortune = 10)
say(fortune = 100)
say(fortune = "whatever")
say(fortune = 7)
say(fortune = 45)
# Clippy
say(fortune = 59, by = "clippy")


library(rmsfact)
say("rms", "rms")


# Using the catfacts API
library(jsonlite)
say("catfact", "cat")
```

Index