Package: wordler (via r-universe)

September 11, 2024

Type Package

Title The 'WORDLE' Game

Version 0.3.1.9001

Description The 'Wordle' game. Players have six attempts to guess a five-letter word. After each guess, the player is informed which letters in their guess are either: anywhere in the word; in the right position in the word. This can be used to inform the next guess. Can be played interactively in the console, or programmatically. Based on Josh Wardle's game <https://www.powerlanguage.co.uk/wordle/>.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/DavidASmith/wordler

Imports crayon

LazyData true

RoxygenNote 7.1.2

Depends R (>= 2.10)

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://davidasmith.r-universe.dev

RemoteUrl https://github.com/davidasmith/wordler

RemoteRef HEAD

RemoteSha 255f012474852a4c6e61a9d94871d3e76e630f43

Contents

`assess_guess` *Assess a guess against the target word*

Description

Assesses the guess in list `game$guess` (index from `game$guess_count`) against the target word in `game$target`.

Usage

```
assess_guess(game)
```

Arguments

`game`	'wordler' game object (as generated by `new_wordler`).

Details

Adds the assessment to the corresponding list item in `game$assess`. This assessment should be considered as how the guesses should be displayed to the user and replicates the behaviour of the WORDLE game (https://www.powerlanguage.co.uk/wordle/).

For each letter in each guess, one of the following assessments are made:

- 'not_in_word' - the letter is not present in the target word (or has already been flagged as 'in_word' earlier in the word).
- 'in_word' - the letter is in the target word. More specifically, the first instance of the letter in the guess present in the word. Subsequent instances are flagged as 'not_in_word'.
- 'in_position' - the letter is in the same position in the target word.

Value

'wordler' game object.

check_guess_hard_mode *Check if current guess complies with hard_mode rules*

Description

Check if current guess complies with hard_mode rules

Usage

```
check_guess_hard_mode(guess, game)
```

Arguments

guess	The guess
game	Wordler game object.

Value

bool

have_a_guess *Submit a guess word to a wordler game object*

Description

If x is a valid guess, it is added to game$guess and assessed against the target word. Increments game$guess_count if a valid guess is made.

Usage

```
have_a_guess(x, game, allowed_words = NULL)
```

Arguments

x	the guess.
game	'wordler' game object (as generated by new_wordler).
allowed_words	a character vector of valid words for the guess. x must be in this vector to be allowed. Defaults to words used by the WORDLE game online (?wordler::wordle_allowed) if not provided.

Value

A 'wordler' game object.

is.wordler *Detects wordler objects*

Description

Detects wordler objects

Usage

```
is.wordler(x, ...)
```

Arguments

x	an R object
...	additional arguments

Value

Returns TRUE if x is a 'wordler' object, otherwise FALSE.

is_guess_correct *Establish if guess is correct and set game state accordingly*

Description

Compares the guess in game$guess (index from game$guess_count) with the corresponding target word in game$target. If the guess is equal to the target, game$game_won and game$game_over are both set to TRUE.

Usage

```
is_guess_correct(game)
```

Arguments

game	'wordler' game object (as generated by new_wordler).

Value

A 'wordler' game object.

keyboards *Keyboard layouts for printing a wordler game at the console.*

Description

A list of keyboard layouts used to show letters known to be not in target word, in the target word, and in the right position in the target word. Each element must be a list having 3 items, each representing a row of a keyboard layout.

Usage

```
keyboards
```

Format

A list of length 1.

Source

https://gist.github.com/cfreshman/cdcdf777450c5b5301e439061d29694c

new_wordler *Constructs a new object of class "wordler"*

Description

Returns a "wordler" object which holds the state of a wordler game as guesses are made. The returned object will have a target word which is selected from the default list unless provided in the `target` argument.

Usage

```
new_wordler(
  target = sample(wordler::wordle_answers, 1),
  game_over = FALSE,
  game_won = FALSE,
  guess_count = 0,
  guess = lapply(1:6, function(x) unlist(strsplit("_____", ""))),
  assess = lapply(1:6, function(x) rep("not_in_word", 5)),
  keyboard = wordler::keyboards$qwerty,
  letters_known_not_in_word = character(0),
  letters_known_in_word = character(0),
  letters_known_in_position = character(0),
  hard_mode = FALSE
)
```

Arguments

target	the target word for the game. Defaults to a random selection from words used by the WORDLE game online (?wordler::wordle_answers) if not provided.
game_over	a logical indicating if the game is over. Defaults to FALSE.
game_won	a logical indicating if the game has been won. In other words, has the target word been correctly guessed.
guess_count	an integer representing the number of guesses made so far. Defaults to 0.
guess	a list (of length 6) of character vectors (each of length 5) representing the guesses of the target word. Each element of the list represents one of six guesses allowed. Each guess defaults to `c("_", "_", "_", "_", "_")` to represent a guess not yet made.
assess	a list (of length 6) of character vectors (each of length 5) representing an assessment of each letter in each guess.
keyboard	a list (of length 3) of character vectors each representing a row of a keyboard layout used to visualise the game by `print()`. Defaults to QWERTY layout.
letters_known_not_in_word	a character vector of letters known not to be in the target word.
letters_known_in_word	a character vector of letters know to be in the target word.
letters_known_in_position	a character vector of letters known to be in the correct position in the target word.
hard_mode	Flag if game is in hard mode

Details

The wordler object is a list which has the following elements:

- `target` - The target word.
- `game_over` - A logical indicating if the game is over. Set to `TRUE` if either the word is correctly guessed, or all guesses are used.
- `game_won` - A logical indicating if the game has been won (target word correctly guessed).
- `guess_count` - The number of guesses made so far.
- `guess` - A list of guesses of the target word.
- `assess` - A list of assessments of the target word. Note that this represents how the letters in each guess should be displayed when printing the game.
- `keyboard` - A list representing the keyboard layout to be used when printing the game state.
- `letters_known_not_in_word` - A vector of letters known not to be in the target word based on guesses made so far.
- `letters_known_in_word` - A vector of letters known to be in the target word based on guesses made so far.
- `letters_known_not_in_word` - A vector of letters known to be in the right position in the target word based on guesses made so far.
- `hard_mode` - A logical indicating if the game is being played in hard_mode. In hard mode any revealed hints must be used in subsequent guesses

Value

An object of class "wordler".

`play_wordler` *Play a game of WORDLE in the console*

Description

Starts an interactive game of WORDLE in the console. Based on WORDLE (https://www.powerlanguage.co.uk/wordle/).

Usage

```
play_wordler(target_words = NULL, allowed_words = NULL, hard_mode = FALSE)
```

Arguments

`target_words`	character vector of potential target words for the game. A word will be randomly selected from this vector as the target word to be guessed. Defaults to words used by the WORDLE game online (?wordler::wordle_answers) if not provided.
`allowed_words`	character vector of valid words for the guess. Guess must be in this vector to be allowed. Defaults to words used by the WORDLE game online (?wordler::wordle_allowed) if not provided.
`hard_mode`	logical flag indicating if hard mode should be used. In hard mode any revealed hints must be used in subsequent guesses

Value

No return value. Starts interactive game in console.

`print.wordler` *Prints a wordler game to the console.*

Description

Prints a wordler game to the console.

Usage

```
## S3 method for class 'wordler'
print(x, ...)
```

Arguments

`x`	'wordler' game object (as generated by `new_wordler`).
`...`	additional arguments

Value

No return value.

`print_instructions` *Prints instructions to play a wordler game in the console*

Description

Prints instructions to play a wordler game in the console

Usage

```
print_instructions()
```

Value

No return value.

`qdap_dict` *All five-letter words from the Nettalk Corpus Syllable Data Set.*

Description

A dataset containing all five-letter words from the Nettalk Corpus Syllable Data Set as returned by qdapDictionaries::dictionaries().

Usage

```
qdap_dict
```

Format

A character vector of length 2488.

Source

https://CRAN.R-project.org/package=qdapDictionaries/

ubuntu_dict *All five-letter words from the Ubuntu dictionary.*

Description

A dataset containing all five-letter words from Ubuntu dictionary '/usr/share/dict/words'.

Usage

```
ubuntu_dict
```

Format

A character vector of length 4594.

Source

https://ubuntu.com/

update_letters_known_in_position *Establish which letters are known to be in the correct position in the target word*

Description

For all items in `game$guess`, establishes the letters which are now known to be in the correct position in the target word. These are present as a character vector in `game$letters_known_in_position` in the returned object.

Usage

```
update_letters_known_in_position(game)
```

Arguments

`game`	'wordler' game object (as generated by `new_wordler`).

Value

A 'wordler' game object.

update_letters_known_in_word

Establish which letters are known to be in the target word

Description

For all items in `game$guess`, establishes the letters which are now known to be in the target word. These are present as a character vector in `game$letters_known_in_word` in the returned object.

Usage

```
update_letters_known_in_word(game)
```

Arguments

`game`	'wordler' game object (as generated by `new_wordler`).

Value

A 'wordler' game object.

update_letters_known_not_in_word

Establish which letters are known to _not_ be in the target word

Description

For all items in `game$guess`, establishes the letters which are now known to not be in the target word. These are present as a character vector in `game$letters_known_not_in_word` in the returned object.

Usage

```
update_letters_known_not_in_word(game)
```

Arguments

`game`	'wordler' game object (as generated by `new_wordler`).

Value

A 'wordler' game object.

wordle_allowed *All words used to validate guesses by the original WORDLE game.*

Description

A dataset containing all words which are used to validate guesses by the original WORDLE game. Note that this does not include the words which can be answers. Theses are held in ?wordle_answers.

Usage

```
wordle_allowed
```

Format

A character vector of length 10657.

Source

https://gist.github.com/cfreshman/cdcdf777450c5b5301e439061d29694c

wordle_answers *All words used as potential answers by the original WORDLE game.*

Description

A dataset containing all words which can be used as answers to the original WORDLE game.

Usage

```
wordle_answers
```

Format

A character vector of length 2315.

Source

https://gist.github.com/cfreshman/a03ef2cba789d8cf00c08f767e0fad7b/

Index