# Package 'covfefe'

January 26, 2026

**Type** Package

**Title** Political Linguistics Toolkit - Gaffes, Phonetics, and Text Transformation

**Version** 1.0.0

**URL** https://github.com/mkirch/covfefe

**BugReports** https://github.com/mkirch/covfefe/issues

**Description** A comprehensive toolkit for political linguistics featuring a museum of famous digital gaffes, phonetic transformation algorithms (Soundex, consonant shifts), QWERTY keyboard geometry for typo simulation, syllable parsing, word blending (portmanteau creation), and text corruption analysis. Originally inspired by the infamous ``covfefe'' tweet of 2017.

**License** GPL (>= 3)

**Encoding** UTF-8

**Depends** R (>= 3.5.0)

**Imports** tokenizers

**RoxygenNote** 7.3.3

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**LazyData** true

**NeedsCompilation** no

**Author** Michael Kirchner [aut, cre]

**Maintainer** Michael Kirchner <michael@kirchner.io>

**Repository** CRAN

**Date/Publication** 2026-01-26 09:50:13 UTC

## Contents

---

`adjacent_key_typo` *Generate adjacent-key typo*

---

## Description

Replaces one or more characters with adjacent keys.

### Usage

```
adjacent_key_typo(word, n_typos = 1L)
```

### Arguments

| | |
|---|---|
| word | Character. Word to typo-fy |
| n_typos | Integer. Number of typos to introduce (default 1) |

### Value

Character. Word with typos

### Examples

```
set.seed(123)
adjacent_key_typo("hello")
```

---

## analyze_gaffe *Analyze a gaffe comprehensively*

---

### Description

Analyze a gaffe comprehensively

### Usage

```
analyze_gaffe(gaffe, intended)
```

### Arguments

| | |
|---|---|
| gaffe | Character. The typo/gaffe |
| intended | Character. The intended word |

### Value

List with comprehensive analysis

### Examples

```
analyze_gaffe("covfefe", "coverage")
```

## blend_score *Score how well a word is a blend of two others*

### Description

Score how well a word is a blend of two others

### Usage

```
blend_score(blend, word1, word2)
```

### Arguments

| | |
|---|---|
| blend | Character. Potential blend |
| word1 | Character. First source word |
| word2 | Character. Second source word |

### Value

Numeric. Score from 0 (poor blend) to 1 (perfect blend)

### Examples

```
blend_score("brunch", "breakfast", "lunch")
```

## blend_words *Blend two words into a portmanteau*

### Description

Combines the beginning of word1 with the end of word2.

### Usage

```
blend_words(word1, word2, overlap = NULL)
```

### Arguments

| | |
|---|---|
| word1 | Character. First word (contributes beginning) |
| word2 | Character. Second word (contributes ending) |
| overlap | Integer. Desired overlap characters (auto-detected if NULL) |

### Value

Character. Blended word

### Examples

```
blend_words("breakfast", "lunch")  # "brunch" or similar
blend_words("refute", "repudiate") # "refudiate" or similar
```

---

## consonant_pairs *Phonetically Similar Consonant Pairs*

---

### Description

Pairs of consonants that are phonetically similar and commonly confused.

### Usage

```
consonant_pairs
```

### Format

A named list mapping consonants to their phonetically similar pairs.

---

## corrupt_text *Corrupt text with specified error type*

---

### Description

Corrupt text with specified error type

### Usage

```
corrupt_text(text, type = "random", position = NULL)
```

### Arguments

| | |
|---|---|
| text | Character. Text to corrupt |
| type | Character. Type of corruption: "keyboard", "phonetic", "double", "truncate", "swap", "omit", "random" |
| position | Integer. Position for positional corruptions (NULL = random) |

### Value

Character. Corrupted text

### Examples

```
set.seed(42)
corrupt_text("hello", type = "keyboard")
corrupt_text("coverage", type = "truncate", position = 3)
```

## covfefy *Covfefy any word.*

### Description

Attempt to produce your own historic typos. This function takes a word and performs the following algorithm thanks to this codegolf thread:

1. Include all characters up through the first vowel in the word. (co)
2. Identifies the next consonant after the first vowel in the word. (v)
3. Changes that consonant to a phonetically similar consonant. (f)
4. Finds the next vowel after that consonant. (e)
5. Combines the two, and repeats. (fefe)
6. Puts it all together: covfefe

### Usage

```
covfefy(str = "coverage")
```

### Arguments

| | |
|---|---|
| str | Character string of word to covfefy. Defaults to "coverage". |

### Examples

```
covfefy("coverage")
covfefy("programming")
covfefy("president")
covfefy("tropical")
```

## covfefySentence *Covfefy any sentence.*

### Description

Calls `covfefy()` and uses rules to decide which, if any, word is covfefefied.

### Usage

```
covfefySentence(
  sent = paste0("Despite the constant negative", " press coverage we are going ",
    "to Make America Great Again"),
  endSentence = TRUE
)
```

### Arguments

| | |
|---|---|
| sent | Character string of sentence to covfefy. Defaults to the famous tweet. |
| endSentence | Boolean where TRUE forces use of punctuation. |

### Examples

```
covfefySentence(paste0("Despite the constant negative press coverage,",
                       "we are going to Make America Great Again"))
```

---

## covfefySpeech *Covfefy any speech.*

---

### Description

Calls `covfefySentences()` and uses rules to decide which, if any, sentences are covfefefied.

### Usage

```
covfefySpeech(
  text = system.file("extdata", "inauguration.txt", package = "covfefe"),
  out = file.path(tempdir(), "covfefe_inauguration.txt")
)
```

### Arguments

| | |
|---|---|
| text | Location of input .txt file. |
| out | Location of output .txt file. |

### Examples

```
covfefySpeech()
```

---

## detect_blend *Detect possible source words for a blend*

---

### Description

Detect possible source words for a blend

### Usage

```
detect_blend(blend, candidates)
```

### Arguments

| | |
|---|---|
| `blend` | Character. The blended word |
| `candidates` | Character vector. Possible source words |

### Value

Character vector. Most likely source words

### Examples

```
detect_blend("brunch", c("breakfast", "lunch", "dinner"))
```

---

## `detect_typo_type` *Detect likely typo type*

---

### Description

Detect likely typo type

### Usage

```
detect_typo_type(typo, intended = NULL)
```

### Arguments

| | |
|---|---|
| `typo` | Character. The typo'd word |
| `intended` | Character. The intended word (if known) |

### Value

List with possible_types and confidence scores

### Examples

```
detect_typo_type("tesy", "test")
```

## gaffes *Political Linguistics Museum Data*

### Description

A data frame containing 13 famous digital gaffes and typos from political figures.

### Usage

```
gaffes
```

### Format

A data frame with 13 rows and 8 variables:

**id** Unique identifier for the gaffe

**original** The intended word or phrase

**gaffe** The actual typo or gaffe

**person** The person who made the gaffe

**year** Year the gaffe occurred

**platform** Platform where the gaffe appeared (Twitter, Truth Social, etc.)

**typo_type** Classification of the typo type

**context** Brief description of the context

### Source

Various news sources and social media archives

## garble_sentence *Garble a sentence with random corruptions*

### Description

Garble a sentence with random corruptions

### Usage

```
garble_sentence(sentence, corruption_rate = 0.3)
```

### Arguments

| | |
|---|---|
| sentence | Character. Sentence to garble |
| corruption_rate | Numeric. Fraction of words to corrupt (0-1) |

### Value

Character. Garbled sentence

### Examples

```
set.seed(42)
garble_sentence("This is a test", corruption_rate = 0.5)
```

---

## get_phonetic_group *Get phonetic group for a consonant*

---

### Description

Get phonetic group for a consonant

### Usage

```
get_phonetic_group(consonant)
```

### Arguments

| | |
|---|---|
| consonant | Character. Single consonant |

### Value

Character. Group name (labial, dental, velar, etc.)

### Examples

```
get_phonetic_group("b")  # "labial"
```

---

## is_keyboard_adjacent *Check if two keys are adjacent*

---

### Description

Check if two keys are adjacent

### Usage

```
is_keyboard_adjacent(key1, key2)
```

### Arguments

| | |
|---|---|
| key1 | Character. First key |
| key2 | Character. Second key |

### Value

Logical

### Examples

```
is_keyboard_adjacent("a", "s")  # TRUE
is_keyboard_adjacent("a", "p")  # FALSE
```

---

## keyboard_mash *Simulate keyboard mash (extra characters)*

---

### Description

Adds random adjacent characters to simulate accidental key presses.

### Usage

```
keyboard_mash(word, n_extra = 3L)
```

### Arguments

| | |
|---|---|
| word | Character. Base word |
| n_extra | Integer. Number of extra characters to add |

### Value

Character. Word with extra characters

### Examples

```
set.seed(456)
keyboard_mash("powerful")  # Might produce "powerfulnnz"
```

---

## museum_by_person *Filter gaffes by person*

---

### Description

Filter gaffes by person

### Usage

```
museum_by_person(pattern)
```

### Arguments

| | |
|---|---|
| pattern | Character. Regex pattern to match person name |

### Value

Data frame of matching gaffes

### Examples

```
museum_by_person("Trump")
```

---

## museum_by_type *Filter gaffes by typo type*

---

### Description

Filter gaffes by typo type

### Usage

```
museum_by_type(type)
```

### Arguments

| | |
|---|---|
| type | Character. Type of typo (e.g., "blend", "keyboard_mash") |

### Value

Data frame of matching gaffes

### Examples

```
museum_by_type("blend")
```

---

## museum_by_year *Filter gaffes by year*

---

### Description

Filter gaffes by year

### Usage

```
museum_by_year(year)
```

### Arguments

| | |
|---|---|
| year | Integer. Year to filter by |

### Value

Data frame of matching gaffes

### Examples

```
museum_by_year(2017)
```

---

## museum_list *List all gaffes in the museum*

---

### Description

List all gaffes in the museum

### Usage

```
museum_list()
```

### Value

Character vector of gaffe IDs

### Examples

```
museum_list()
```

---

## museum_lookup *Look up a specific gaffe*

---

### Description

Look up a specific gaffe

### Usage

```
museum_lookup(id)
```

### Arguments

| | |
|---|---|
| id | Character. The gaffe ID (e.g., "covfefe", "potatoe") |

### Value

List with gaffe details

### Examples

```
museum_lookup("covfefe")
```

## museum_random *Get a random gaffe from the museum*

### Description

Get a random gaffe from the museum

### Usage

```
museum_random()
```

### Value

List with gaffe details

### Examples

```
set.seed(42)
museum_random()
```

## onset_nucleus_coda *Parse syllable into onset, nucleus, coda*

### Description

Parse syllable into onset, nucleus, coda

### Usage

```
onset_nucleus_coda(syllable)
```

### Arguments

| | |
|---|---|
| syllable | Character. Single syllable |

### Value

List with onset, nucleus, coda components

### Examples

```
onset_nucleus_coda("cat")  # list(onset="c", nucleus="a", coda="t")
```

## phonetic_distance *Calculate phonetic distance between words*

### Description

Uses Soundex codes to measure phonetic similarity.

### Usage

```
phonetic_distance(word1, word2)
```

### Arguments

| | |
|---|---|
| word1 | Character. First word |
| word2 | Character. Second word |

### Value

Numeric. Distance (0 = identical, higher = more different)

### Examples

```
phonetic_distance("coverage", "covfefe")
```

## phonetic_groups *Phonetic Consonant Groups*

### Description

Consonants grouped by place of articulation.

### Usage

```
phonetic_groups
```

### Format

A named list with groups: labial, dental, velar, sibilant, liquid, glide.

## phonetic_shift *Shift consonant to phonetically similar consonant*

### Description

Shift consonant to phonetically similar consonant

### Usage

```
phonetic_shift(word, target)
```

### Arguments

| | |
|---|---|
| word | Character. Word to transform |
| target | Character. Consonant to shift (if found) |

### Value

Character. Transformed word

### Examples

```
phonetic_shift("coverage", "v")  # "coferage"
```

## qwerty_adjacency *QWERTY Keyboard Adjacency Data*

### Description

A list containing adjacent keys for each letter on a QWERTY keyboard.

### Usage

```
qwerty_adjacency
```

### Format

A named list with 26 elements (a-z), each containing a character vector of adjacent key letters.

## qwerty_adjacent *Get adjacent keys on QWERTY keyboard*

### Description

Get adjacent keys on QWERTY keyboard

### Usage

```
qwerty_adjacent(key)
```

### Arguments

| | |
|---|---|
| key | Character. Single lowercase letter |

### Value

Character vector of adjacent keys

### Examples

```
qwerty_adjacent("f")
```

## qwerty_distance *Calculate QWERTY distance between two keys*

### Description

Uses Euclidean distance based on key positions.

### Usage

```
qwerty_distance(key1, key2)
```

### Arguments

| | |
|---|---|
| key1 | Character. First key (single lowercase letter) |
| key2 | Character. Second key (single lowercase letter) |

### Value

Numeric distance

### Examples

```
qwerty_distance("a", "s")  # Adjacent = 1
qwerty_distance("a", "p")  # Far apart
```

## qwerty_layout *QWERTY Keyboard Layout Data*

### Description

A list containing row and column positions for each letter on a QWERTY keyboard.

### Usage

```
qwerty_layout
```

### Format

A named list with 26 elements (a-z), each containing a named numeric vector with 'row' and 'col' positions.

## soundex *Calculate Soundex code*

### Description

American Soundex algorithm for phonetic encoding.

### Usage

```
soundex(word)
```

### Arguments

| | |
|---|---|
| word | Character. Word to encode |

### Value

Character. 4-character Soundex code

### Examples

```
soundex("Robert")     # "R163"
soundex("Rupert")     # "R163" (same as Robert)
soundex("Washington") # "W252"
```

## soundex_map *Soundex Encoding Map*

### Description

Mapping of consonants to their Soundex digit codes.

### Usage

```
soundex_map
```

### Format

A named character vector mapping consonants to digits 1-6.

## suggest_corrections *Suggest corrections for a typo*

### Description

Suggest corrections for a typo

### Usage

```
suggest_corrections(typo, dictionary = NULL)
```

### Arguments

| | |
|---|---|
| typo | Character. The typo'd word |
| dictionary | Character vector. Optional custom dictionary |

### Value

Character vector of suggested corrections

### Examples

```
suggest_corrections("tesy")
```

## syllabify *Split word into syllables*

### Description

Uses a simplified vowel-based algorithm.

### Usage

```
syllabify(word)
```

### Arguments

| | |
|---|---|
| word | Character. Word to syllabify |

### Value

Character vector of syllables

### Examples

```
syllabify("hello")     # c("hel", "lo")
syllabify("coverage")  # c("cov", "er", "age")
```

## syllable_count *Count syllables in a word*

### Description

Count syllables in a word

### Usage

```
syllable_count(word)
```

### Arguments

| | |
|---|---|
| word | Character. Word to count |

### Value

Integer. Number of syllables

### Examples

```
syllable_count("hello")  # 2
```

## syllable_swap *Swap syllables between two words*

### Description

Swap syllables between two words

### Usage

```
syllable_swap(word1, word2, position = 1L)
```

### Arguments

| | |
|---|---|
| word1 | Character. First word |
| word2 | Character. Second word |
| position | Integer. Which syllable to swap (1-indexed) |

### Value

Character vector of length 2 with swapped words

### Examples

```
syllable_swap("coverage", "president", 1)
```

## vowel_shift *Shift vowels in a word*

### Description

Shift vowels in a word

### Usage

```
vowel_shift(word, from, to)
```

### Arguments

| | |
|---|---|
| word | Character. Word to transform |
| from | Character. Vowel to replace |
| to | Character. Replacement vowel |

### Value

Character. Transformed word

## Examples

```
vowel_shift("test", "e", "i")  # "tist"
```

---

`vowel_shifts` *Vowel Shift Mappings*

---

## Description

Possible vowel shifts inspired by historical sound changes.

## Usage

```
vowel_shifts
```

## Format

A named list mapping vowels to possible shift targets.

# Index