It’s October, time for spooky Twitter names! If you’re on this social  
media platform, you might have noticed some of your friends switching  
their names to something spooky and punny. Last year I was “Maelstrom  
Salmon”, which I find scary but is arguably not that funny. Anyhow, what  
if you want to switch your name but have no inspiration? In this post,  
we shall explore R’s abilities to help us with that with the help of  
webscraping, phonetic spelling and string distance algorithms, and the  
magic of randomness!

My strategy for spooky name generation will be to replace each part of a  
name (e.g. first and last names) with the phonetically closest  
Halloween-related words. Therefore, the technical challenges here were  
to find a list of Halloween-related words, and to measure *phonetical*  
string distance.

**Getting spooky words**

There were two less useful sections on the page, with French and Spanish  
words, that I very inelegantly removed after looking up their div ID in  
the page source.

Library(Polite)

Library(rcorpora)

# scrape

page <- polite::scrape(session)

## No encoding supplied: defaulting to UTF-8.

# remove the two sections I don't want

French <- xml2::xml\_find\_all(page,

xpath = "//div[@id='mod\_37589248']")

xml2::xml\_remove(French)

Spanish <- xml2::xml\_find\_all(page,

xpath = "//div[@id='mod\_37589143']")

xml2::xml\_remove(Spanish)

# extract the words

words <- xml2::xml\_find\_all(page,

xpath = "//td[contains(@class, 'tableCell')]")

words <- xml2::xml\_text(words)

head(words)

## [1] "All Hallows' Eve\n\t" "lantern\n\t" "prank\n\t"

## [4] "costume\n\t" "make-believe\n\t" "sweets\n\t"

I cleaned the words a bit. Some of them were not *words* so I split  
them. I hesitated between this solution and keeping only one-word  
phrases… I’m still not sure it was the best decision!

words <- stringr::str\_remove\_all(words, "\\\n\\\t")

words <- trimws(words)

words <- tokenizers::tokenize\_words(words)

words <- unlist(words[lengths(words) > 0])

words <- words[!stringr::str\_detect(words, "[0-9]")]

head(words)

## [1] "all" "hallows" "eve" "lantern" "prank" "costume"

I obtained 258 words.

**Matching names and spooky words**

Once I had this vector of Halloween-related words, all I needed was a  
way to compute a phonetical distance between each of them and a name. It  
turned out more complicated than I thought! The soundex method of the  
stringdist package  
returned 1 or 0 only, so I set out to search another package supporting  
phonetical comparison. I found the  
phonics package that has a  
paper in JOSS, that implements more algorithms translating strings to  
phonetic “codes”. I ended up using the phonics::nysiis() function,  
corresponding to the New York State Identification and Intelligence  
System phonetic algorithm. That package also has an algorithm suitable  
for German, via phonics::cologne(), and one for French, via  
phonics::statcan(), but I haven’t explored that further.

phonics::nysiis("pain")

## [1] "PAN"

phonics::nysiis("pane")

## [1] "PAN"

phonics::nysiis("train")

## [1] "TRAN"

I was at a loss as to how best compare output from phonics::nysiis()  
and very hackily used… stringdist::stringdist() with a default method!  
Below is the function finding the closest match for any name part.

spookify\_word <- function(word, seed = 42, words){

phon <- phonics::nysiis(words)

dist <- stringdist::stringdist(phonics::nysiis(word),

phon)

set.seed(seed)

sample(words[dist == min(dist)], 1)

}

spookify\_word("Jane", words = words)

## [1] "bones"

So spookify\_word() samples a word among the closest matches… that  
aren’t necessarily very close to the original name. But that’s fine!

**Mixing it all in a function**

I finally created a function taking a whole name, say “Maëlle Salmon” or  
“Ada Colau Ballano”, tokenizes it into words thanks to the  
tokenizers package, gets the  
closest match for each part and then collapses everything together,  
capitalizing first letters thanks to  
snakecase… before announcing  
the result with the cool cowsay  
package! I made sure to only use animals that are associated with  
Halloween.

spookify <- function(name, seed,

words){

name\_parts <- unlist(tokenizers::tokenize\_words(name))

spook\_names <- purrr::map\_chr(name\_parts,

spookify\_word,

seed, words)

spook\_name <- paste(spook\_names, collapse = " ")

spook\_name <- snakecase::to\_upper\_camel\_case(spook\_name,

sep\_out = " ")

set.seed(seed)

animal <- sample(c("owl", "spider",

"pumpkin", "ghost",

"bat"), 1)

cowsay::say(what = paste0("Byebye ", name, "\n",

"You are now ", spook\_name),

by = animal)

}

And because I didn’t want to make fun of anyone else than me, I created  
7 fake names using the charlatan package and ran spookify on them!

set.seed(42)

names <- charlatan::ch\_name(n = 7)

purrr::walk2(names, 1:length(names),

spookify, words = words)

## Colors cannot be applied in this environment :( Try using a terminal or RStudio.

##

##

## -----

## Byebye Tyrik Graham PhD

## You are now Trick Grinning Giant

## ------

## \

## \

## |

## |

## |

## \_\_

## | / \ |

## \\_\\ //\_/

## .'/()\'.

## \\ // [nosig]

##

## Colors cannot be applied in this environment :( Try using a terminal or RStudio.

##

## -----

## Byebye Jeremy Rippin

## You are now Horrify Rotten

## ------

## \

## \

## \

## /\\_\_\_/\

## {o}{o}|

## \ v /|

## | \ \

## \\_\_\_/\_/ [ab]

## | |

## Colors cannot be applied in this environment :( Try using a terminal or RStudio.

##

## -----

## Byebye Margarett Purdy

## You are now Boogers Party

## ------

## \

## \

## \

## /\\_\_\_/\

## {o}{o}|

## \ v /|

## | \ \

## \\_\_\_/\_/ [ab]

## | |

## Colors cannot be applied in this environment :( Try using a terminal or RStudio.

##

##

## -----

## Byebye Miss Tilla Funk DDS

## You are now Ooze Vil Fangs Dead

## ------

## \

## \

## \_\_\_

## \_\_\_)\_\_|\_

## .-\*' '\*-,

## / /| |\ \

## ; /\_| |\_\ ;

## ; |\ /| ;

## ; | ''--...--'' | ;

## \ ''---.....--'' /

## ''\*-.,\_\_\_\_\_\_\_,.-\*' [nosig]

##

## Colors cannot be applied in this environment :( Try using a terminal or RStudio.

##

##

## -----

## Byebye Wilton Carroll IV

## You are now Cartoon Cruella Eve

## ------

## \

## \

## |

## |

## |

## \_\_

## | / \ |

## \\_\\ //\_/

## .'/()\'.

## \\ // [nosig]

##

## Colors cannot be applied in this environment :( Try using a terminal or RStudio.

##

##

## -----

## Byebye Tevin Conn

## You are now Coffin Bones

## ------

## \

## \

## .-.

## (o o)

## | O \

## \ \

## `~~~' [nosig]

##

## Colors cannot be applied in this environment :( Try using a terminal or RStudio.

##

##

## -------------

## Byebye Dr. Ballard Langworth III

## You are now Dish Blood Cemetery Dish

## --------------

## \

## \

## \

## \_\_.--'\ \.\_\_./ /'--.\_\_

## \_.-' '.\_\_.' '.\_\_.' '-.\_

## .' '.

## / \

## | |

## | |

## \ .---. .---. /

## '.\_ .' '.''. .''.' '. \_.'

## '-./ \ / \.-'

## ''mrf

I found the results not fantastic, but not too bad either! I especially  
liked “Boogers Party”!

**Mucus, anyone?**

In conclusion, in this post I showed how to build a corpus of  
Halloween-related words by responsibly webscraping, and how to more or  
less hackily compare strings based on their pronunciation. I obviously  
tried the function on my own name (without the accent, useless in this  
case) and wasn’t too happy…

spookify("X", 42, words)

## Colors cannot be applied in this environment :( Try using a terminal or RStudio.

##

##

## -------------

## Byebye X

## You are now Mucus Bulging

## --------------

## \

## \

## \

## \_\_.--'\ \.\_\_./ /'--.\_\_

## \_.-' '.\_\_.' '.\_\_.' '-.\_

## .' '.

## / \

## | |

## | |

## \ .---. .---. /

## '.\_ .' '.''. .''.' '. \_.'

## '-./ \ / \.-'

## ''mrf

But then, at least, it *is* scary.