DS in Context

Ethan Haley

3/4/2021

```r
people <- c('ZuJe', 'ZegRcad', 'SliaDne',
              'SollKnn', 'SnhTg', 'SnhRmia', 'SamDea',
              'SsdaaRtih', 'SfrZcay', 'RdiuzEcie',
              'ReeSme', 'OesHny', 'NtTiht',
              'MsoDne', 'MyrCar', 'MLuhiEa',
              'MrieGbila','LccMthw', 'IlmMla',
              'IpltMcal', 'HmelJsu', 'HrcEi',
              'HrasBa', 'HlyEhn', 'GedagJra',
              'GnroMrAeada', 'GtcPdo',
              'FaknegTlr', 'FruoCril', 'CseCsada',
              'CnolJsp', 'CniSa', 'CaVc',
              'CmoGbil', 'BrsvDiry', 'AaaoEtbn')
length(people)
```

```
## [1] 36
```

```r
# Input a word of mixed-case letters, output a vector of 1-52 equivs,
##  optionally lower-cased with second parameter
string2nums <- function(string, makeLower = FALSE) {
  chars <- unlist(strsplit(string, ''))
  nums <- lapply(chars, function(ch){match(ch, c(letters, LETTERS))})
  nums <- as.numeric(nums)
  if (makeLower) {nums <- nums %% 26}
  nums
}
string2nums('HlyEhn', makeLower = T)
```

```
## [1]  8 12 25  5  8 14
```

```r
# how far apart are 2 letters on a wheel?
#  Inputs are 2 integers from 1-26, representing a-z, or A-Z
#  Image taken from https://kodlogs.com/blog/618/alphabet-wheel
cycleDist <- function(letternum1, letternum2) {
  d <- letternum1 - letternum2
  if (d > 0) {
    d <- min(d, letternum2 + 26 - letternum1)
  } else if (d < 0) {
    d <- min(-d, letternum1 + 26 - letternum2)
  }
  d
}
c(cycleDist(1,1), cycleDist(3,22), cycleDist(48,29), cycleDist(5,19))
```

```
## [1]  0  7  7 12
```

manhattan, jaccard, cosine, levenshtein / keyboard, tfidf

```r
#jaccard can compare strings
jaccard <- function(string1, string2) {
  set1 <- unique(unlist(strsplit(string1, '')))
  set2 <- unique(unlist(strsplit(string2, '')))
  1 - length(intersect(set1, set2)) / length(union(set1, set2))
}
jaccard('carjacked', 'jaccard')
```

```
## [1] 0.2857143
```

$d_{Jaccard}(X,Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$

$d_{cosine}(X,Y) = 1 - \frac{X \cdot Y}{||X||_2 \cdot ||Y||_2}$

$< 3, 4, 5 >$

Levenshtein metric – related to sequence alignment of genes

Tim Roughgarden's dynamic programming video for sequence alignment, on Coursera

```r
cosine <- function(string1, string2, len = 52) {
  nums1 <- string2nums(string1)
  nums2 <- string2nums(string2)
  vec1 <- rep(0, len)
  vec2 <- rep(0, len)
  for (n in nums1) {
    vec1[n] <- vec1[n] + 1
  }
  for (n in nums2) {
    vec2[n] <- vec2[n] + 1
  }
  dot <- sum(vec1 * vec2)
  mags <- sqrt(sum(vec1 * vec1)) * sqrt(sum(vec2 * vec2))
  1 - dot / mags
}
cosine('condescension', 'cosine') # length adds similarity, if
```

```
## [1] 0.05719096
```

```r
jaccard('condescension', 'cosine') # only get "credit" for one instance of each letter
```

```
## [1] 0.1428571
```

```r
adist('condescension', 'cosine')[[1]]  #Levenshtein metric / edit distance, from utils pkg
```

```
## [1] 9
```

```r
adist('jaccard', 'carjacked')[[1]]
```

```
## [1] 6
```

```r
cosine('the', 'car')
```

```
## [1] 1
```

```r
cosine('the', 'teeth')
```

```
## [1] 0.03774955
```

```r
adist('the', 'car')[[1]]
```

```
## [1] 3
```

```r
drop(attr(adist("the", "teeth", counts = TRUE), "counts"))
```

```
## ins del sub
##   2   0   1
```

```r
adist("The", "teeth", ignore.case = TRUE)[[1]]
```

```
## [1] 3
```

```r
# keyboard neighbors as weightings for penalties
neighbors <- list(c('q','w','s','z'), c('v', 'g', 'h', 'n'),
                  c('x', 'd', 'f', 'v'), c('e', 'r', 'f', 'c', 'x', 's'),
                  c('w', 's', 'd', 'r'),      c()     , c('a', 's', 'x'))
```

```r
# application to gene-similarity algorithms
#  "A" isn't just 1 away from "G", it's 2.  So penalize substitution 2x as much.
adist('nine', 'mice')[[1]]
```

```
## [1] 2
```

```r
drop(attr(adist("nine", "mice", counts = TRUE), "counts"))
```

```
## ins del sub
##   0   0   2
```

```
adist('nine', 'mice', costs = c(i=1, d=1, s=2))[[1]]
```

```
## [1] 4
```

```
drop(attr(adist("nine", "mice", costs = c(i=1, d=1, s=2), counts = TRUE), "counts"))
```

```
## ins del sub
##   2   2   0
```

not penalizing substitutions:

n i n e

m i c e

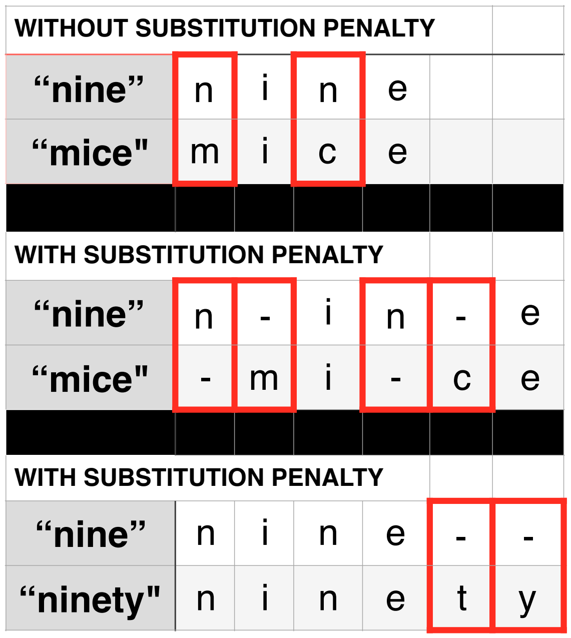
penalizing substitutions:

- n i - n e

m - i c - e

Needleman-Wunsch algorithm

```
knitr::include_graphics('nineMicePenaltiesHighlighted.pdf')
```



WITHOUT SUBSTITUTION PENALTY						
“nine”	n	i	n	e		
“mice"	m	i	c	e		
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

WITH SUBSTITUTION PENALTY						
“nine”	n	-	i	n	-	e
“mice"	-	m	i	-	c	e
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

WITH SUBSTITUTION PENALTY						
“nine”	n	i	n	e	-	-
“ninety"	n	i	n	e	t	y

1