**A COVID19 lexicon**

library(tidyverse)

tweets\_dir <- '/home/jtimm/jt\_work/GitHub/git\_projects/us\_lawmaker\_tweets\_2020/'

covid\_dir <- '/home/jtimm/jt\_work/GitHub/git\_projects/A-covid19-lexicon/'

setwd(covid\_dir)

dictionary <- readxl::read\_xlsx ('covid\_glossary\_w\_variants.xlsx') %>%

filter(category != 'race-ethnicity') %>%

ungroup()

I have collated some COVID19-related terms from a few resources, most notably, this Yale Medicine [glossary](https://www.yalemedicine.org/stories/covid-19-glossary/). Per this resource, each term has been categorized as one of the following:

unique(dictionary$category)

## [1] "cv" "interventions" "medical\_response"

## [4] "prevention" "socio-political" "spread\_of\_disease"

## [7] "transmission"

As I have added terms, I have tried to fit them within this classification framework. I have also added a socio-political category to capture some of the civil-liberties-based rhetoric/protesting happening in the US in response to stay-at-home orders, as well as stimulus legislation, etc. A good start, but could certainly be developed.

The table below illustrates the structure of the vocabulary for two COVID19-related concepts: ANTIVIRAL and HAND-HYGIENE. So, the descriptor\_name column represents the higher-level concept; term\_name column reflects the different (inflectional or orthographical) ways the concept can manifest in text. The actual form of the descriptor/concept is arbitrary.

So, as we move towards identifying/extracting COIV19-related terms from Twitter text, this vocabulary gives us the option to aggregate over terms to the higher-level concept (or descriptor). Some academic fields refer to this as the process of normalization.

dictionary %>%

filter(descriptor\_name %in% c('antiviral', 'hand-hygiene')) %>%

group\_by(category, descriptor\_name) %>%

summarize(term\_names = paste0(term\_name, collapse = ' | ')) %>%

DT::datatable(rownames = FALSE, options = list(dom = 't'))

**Congressional Twitter Corpus (2020)**

Again, our data set is a Twitter corpus comprised of all tweets generated by the **535 voting members of the US Congress** during the second session of the 116th Congress.

setwd(paste0(tweets\_dir, 'tweets'))

tweets <- readxl::read\_xlsx('us\_lawmaker\_tweets\_full\_2020-05-26.xlsx') %>%

mutate(created\_at = as.Date(created\_at, format = "%Y-%B-%d"))

Corpus composition:

data.frame(tweets = format(nrow(tweets), big.mark = ','),

tokens = format(sum(tokenizers::count\_words(tweets$text)),

big.mark = ',')) %>%

knitr::kable()

| **tweets** | **tokens** |
| --- | --- |
| 157,420 | 5,650,645 |

Next, we quickly grab some details about US lawmakers from [the united states project](https://theunitedstates.io/). The Twitter corpus and lawmaker detail data sets can then be joined via Twitter handle.

leg\_dets <- '<https://theunitedstates.io/congress-legislators/legislators-current.csv>'

twitters <- read.csv((url(leg\_dets)), stringsAsFactors = FALSE) %>%

rename (state\_abbrev = state, district\_code = district)

tweets1 <- tweets %>%

mutate(screen\_name = toupper(screen\_name)) %>%

left\_join(twitters %>%

mutate(twitter = toupper(twitter)),

by = c('screen\_name' = 'twitter'))

**Some sample tweets:**

**text2vec framework for NLP ::**

text2vec is a beast of a [text analysis R library](http://text2vec.org/). Here, we walk-through the building of some common text structures relevant to many downstream applications – using our congressional Twitter corpus. As text2vec implements R6 objects (a mystery to me), the framework is a bit funky. So, we present some hacks, etc. here, specifically for working with multi-word expressions – in the larger context of building document-term matrices, term-co-occurrence matrices, GloVe models & co-occurrence-based graph structures.

With the ultimate goal of investigating (1) some historical- and party-affiliation-based variation in the use of COVID19-related terms on Twitter, and (2) the conceptual relatedness of COVID19-related terms

**Tokens & tokenizers**

text2vec, like other text analysis frameworks, operates on a token object, which for a single document/tweet looks like the following:

tokenizers::tokenize\_ptb(tweets$text[2], lowercase = TRUE)

## [[1]]

## [1] "ohioans" ":" "request"

## [4] "a" "mail-in" "ballot"

## [7] "today" "from" "the"

## [10] "secretary" "of" "state"

## [13] "to" "ensure" "your"

## [16] "vote" "is" "counted"

## [19] "in" "ohio" "'s"

## [22] "primary" "election." "the"

## [25] "deadline" "to" "postmark"

## [28] "your" "ballot" "is"

## [31] "monday." "https" ":"

## [34] "//[t.co/gkdascowqc](http://t.co/gkdascowqc)"

Tokenization, even for English, is a non-trivial task. The tokenize\_ptb function from the tokenizers package is pretty good (which is based on the Penn Treebank model). But there are still two instances above, eg, in which sentence-final punctuation is not tokenized: election. & monday.. So, when we go to build word-level models, election & election., eg, will be treated distinctly.

This bothers me. The code below sorts this and other issues out. Resulting/re-built text can then be fed to any simple space-based tokenizer, and things will be clean. *Tokenize > clean tokens > rebuild text > re-tokenize*.

## tokenizer --

t1 <- tokenizers::tokenize\_ptb(tweets$text, lowercase = TRUE)

## Remove punct

t2 <- lapply(t1, gsub,

pattern = '([a-z0-9])([[:punct:]])',

replacement = '\\1 \\2')

t3 <- lapply(t2, gsub,

pattern = '([[:punct:]])([a-z0-9])',

replacement = '\\1 \\2')

t4 <- lapply(t3, paste0, collapse = ' ')

## Re-build

tweets$word\_text <- unlist(t4)

**Multi-word expressions & controlled vocabularies**

Units of meaning often (ie, almost always) span multiple words and multiple grammatical categories. Here we briefly consider some supervised approaches to tricking tokenizers (and specifically text2vec) into treating a controlled vocabulary of multi-word expressions as single-units-of-meaning.

**§ Some multi-word hacks**

The spelling & inflectional variants of the COVID19-related concept FLATTEN THE CURVE are presented below:

| **term\_names** |
| --- |
| flatten the curve | flatten\_the\_curve | flatten-the-curve | flattening the curve | flattening\_the\_curve | flattening-the-curve | flatteningthecurve | flattenthecurve |

So, if a lawmaker on Twitter refers to the concept FLATTEN THE CURVE as flattenthecurve, without any spaces (& presumably prefixed with a hash tag), a space-based (or word-based) tokenizer will do right by the analyst investigating multi-word expressions. The same goes for flatten-the-curve and flatten\_the\_curve.

The form flatten the curve, however, will be tokenized as flatten and the and curve. Which is not helpful. Basically, we want to phrasify these three individual tokens as a single token. Such that in downstream applications, flatten the curve and flattenthecurve, eg, are (or can be) treated as instantiations of the same conceptual category.

The Collocations function/model from the text2vec package enables an unsupervised approach to identifying multi-word expressions, and results can be used to update token objects such that flatten the curve becomes flatten-the-curve. If flatten + the + curve is identified as an expression per the model.

Here, however, we are interested in a supervised (or controlled) approach, ie, we have our own multi-word lexicon of COVID19-related terms that we want phrasified. text2vec does not provide a straightforward way to do this. So, here we present a simple (albeit extended) hack.

multi\_word\_expressions <- subset(dictionary, grepl(' ', term\_name))

sep = ' '

mas\_que\_dos <- subset(multi\_word\_expressions, grepl(' [a-z0-9]\* ', term\_name))

First: text2vec::Collocations builds out phrases in a piecemeal fashion. Long story short: in order to identify (or phrasify) flatten the curve as a multi-word expression, it must first identify (or phrasify), eg, flatten the. Then flatten-the and curve can be phrasified as flatten-the-curve. So, for multi-word expressions > 2, we have to build out some component parts. Some multi-word expressions in the COVID19 vocabulary >2 words:

## [1] "flattening the curve" "drive thru tests"

## [3] "personal protective equipment" "flatten the curve"

## [5] "front line worker" "great american comeback"

## [7] "global economic cirsis" "return to work"

## [9] "high risk population" "god bless america"

A simple function for extracting component 2-word phrases from multi-word expressions >2:

new\_two\_grams <- lapply(mas\_que\_dos$term\_name, function(x) {

regmatches(x,

gregexpr("[^ ]+ [^ ]+", # sep = ' '

x,

perl=TRUE)

)[[1]] }) %>%

unlist() %>%

unique()

A look at the “pieces” of our mutli-word expressions composed of more than two words:

## [1] "sars cov" "drive through" "drive thru"

## [4] "personal protective" "flatten the" "flattening the"

## [7] "front line" "high risk" "shelter in"

## [10] "long term" "stay at" "home order"

## [13] "home orders" "wash your" "wear a"

## [16] "work from" "working from" "dont bankrupt"

## [19] "global economic" "global health" "god bless"

## [22] "great american" "made in" "open up"

## [25] "paycheck protection" "re open" "return to"

## [28] "person to" "person transmission"

Then we add these “pieces” to the full multi-word portion of the COVID19 lexicon.

multi\_word\_expressions\_replace <- gsub(' ', sep, multi\_word\_expressions$term\_name)

multi\_word\_expressions\_replace <- c(multi\_word\_expressions\_replace,

new\_two\_grams )

**§ Some text2vec primitives**

Before we can dupe text2vec into phrasifying our multi-word COVID19 terms, we first need to build two basic text2vec (data) structures: an itoken object (or iterator) & a vocabulary object. The former containing (among other things) a generic tokens object. Again, see [this vignette](http://text2vec.org/) for more technical details. Regardless of your text2vec objectives, these will (almost) always be your first two opening moves.

mo <- text2vec::itoken(tweets$word\_text,

preprocessor = tolower,

tokenizer = text2vec::space\_tokenizer,

n\_chunks = 1,

ids = tweets$status\_id)

vocab <- text2vec::create\_vocabulary(mo, stopwords = character(0)) #tm::stopwords()

Then we build a skeleton Collocations model per code below. But we never actually run the model.

model <- text2vec::Collocations$new(vocabulary = vocab, sep = sep)

Instead, all we want to do is assign the parameter model$.\_\_enclos\_env\_\_$private$phrases our list of multi-word expressions.

model$.\_\_enclos\_env\_\_$private$phrases <- multi\_word\_expressions\_replace

Using this dummy Collocations model, we then transform the itoken object built above. Here, transform means updating the token object to account for multi-word expressions.

it\_phrases <- model$transform(mo)

term\_vocab <- text2vec::create\_vocabulary(it\_phrases)

term\_vocab1 <- text2vec::prune\_vocabulary(term\_vocab, term\_count\_min = 2)

## HACK

ats <- attributes(term\_vocab1)

term\_vocab2 <- subset(term\_vocab1, grepl('^[A-Za-z]', term) & nchar(term) > 2)

t2v\_vocab <- term\_vocab2

attributes(t2v\_vocab) <- ats

#egs <- it\_phrases$nextElem()$tokens

#egs1 <- lapply(egs, paste0, collapse = ' ')

And now we can investigate frequencies for all forms included in the congressional Twitter corpus, including (but not limited to) our multi-word expressions.

term\_freqs <- term\_vocab2 %>%

left\_join(dictionary , by = c('term' = 'term\_name'))

descriptor\_freq <- term\_freqs %>%

group\_by(category, descriptor\_name) %>%

summarize(term\_freq = sum(term\_count)) %>%

filter(![is.na](http://is.na)(descriptor\_name))

**Some relative frequencies** for spelling & lexical variants for a sample of multi-word expressions from the COVID19 lexicon.

term\_freqs %>%

filter(descriptor\_name %in% c('social-distancing', 'front-line-workers',

'flatten-the-curve')) %>%

arrange(desc(term\_count)) %>%

mutate(tf = paste0(term, ' (', term\_count, ')')) %>%

group\_by(descriptor\_name) %>%

summarize(relative = paste0(tf, collapse = ' | ')) %>%

DT::datatable(rownames = FALSE, options = list(dom = 't'))

So, the trickier/hackier part is complete. The text2vec vocabulary object now recognizes the multi-word expressions in our COVID19 lexicon as single units of meaning. And we can carry on.

**DTMs & prevalence of COVID19-related concepts**

**§ Document-term matrix**

DTMs can be leveraged in any number of ways, for any number of downstream linguistic applications. Here, we want to use the DTM to understand variation in the use of COVID19-related concepts as a function of time and political affiliation – the former a feature of the tweet; the latter a feature of the tweeter, ie, the US lawmaker.

The text2vec::create\_dtm function requires an additional object, a vocab\_vectorizer, which is derived from the vocabulary object discussed/created above. The vocab\_vectorizer operates on the phrasified token object contained in it\_phrases. Using my lexvarsdatr package, we then convert the DTM to a long data frame.

vectorizer <- text2vec::vocab\_vectorizer(t2v\_vocab)

dtm0 <- text2vec::create\_dtm(it\_phrases, vectorizer)

dtm1 <- lexvarsdatr::lvdr\_get\_closest(dtm0) ## other ways --

colnames(dtm1) <- c('doc\_id', 'term\_name', 'count')

dtm2 <- dtm1 %>% inner\_join(dictionary)

**§ Historical prevalences**

Per this new data structure, we can now add tweet metadata (here, date of creation) as well as tweeter details, eg, name, party affiliation, gender, etc.

dtm3 <- dtm2 %>%

## Aggregate -- to descriptor\_name

group\_by(doc\_id, descriptor\_name, category) %>%

summarize(count = sum(count)) %>%

ungroup() %>%

left\_join(tweets %>% select(status\_id, screen\_name, created\_at) %>%

mutate(screen\_name = toupper(screen\_name)), by = c('doc\_id' = 'status\_id')) %>%

left\_join(twitters %>%

select(full\_name, type:district\_code, twitter, party) %>%

mutate(twitter = toupper(twitter)), by = c('screen\_name' = 'twitter'))

We can now calculate the **daily rate of reference** for each concept included in the COVID19 lexicon. Here, the rate of reference for concept **X** is calculated as the # of tweets referring to concept **X** per 1K total tweets (for a given day).

for\_plot\_historical <- dtm3 %>%

group\_by(created\_at) %>%

mutate(daily\_total = length(unique(doc\_id))) %>%

group\_by(descriptor\_name, created\_at, daily\_total) %>%

summarize (n = n()) %>%

mutate(per\_k\_tweets = round(n/daily\_total \* 1000, 3)) %>%

ungroup()

# mutate(cumulative\_n = cumsum(n)) %>%

# complete(created\_at = seq.Date(from = min(created\_at),

# to = max(created\_at),

# by="day"),

# descriptor\_name) %>% fill(cumulative\_n)

for\_plot\_historical$per\_k\_tweets[[is.na](http://is.na)(for\_plot\_historical$per\_k\_tweets)] <- 0

**§ Historical plot**

The plot below illustrates daily rates of reference to several COVID19-related concepts since the beginning of March. Black lines represent 7-day moving averages.

keeps <- c('cares-act', 'flatten-the-curve',

'stay-at-home', 'social-distancing',

'essential-workers', 'face-mask',

'paycheck-protection-program', 'first-responders',

'ventilator')

###

for\_plot\_historical %>%

filter(descriptor\_name %in% keeps &

created\_at > '2020-03-01') %>%

mutate(moving\_n = zoo::rollmean(per\_k\_tweets, k = 7, fill = NA)) %>%

ggplot() +

geom\_bar(aes(x = created\_at,

y= per\_k\_tweets, #cumulative\_n,

fill = descriptor\_name),

stat="identity") +

#geom\_bar(stat="identity") +

geom\_line(aes(x = created\_at,

y = moving\_n),

color = 'black',

#linetype = 2,

size = .75) +

theme\_minimal() +

ggthemes::scale\_fill\_stata() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1))+

scale\_x\_date(date\_breaks = '1 week', date\_labels = "%b %d") +

theme(legend.position = 'none',

legend.title = element\_blank()) +

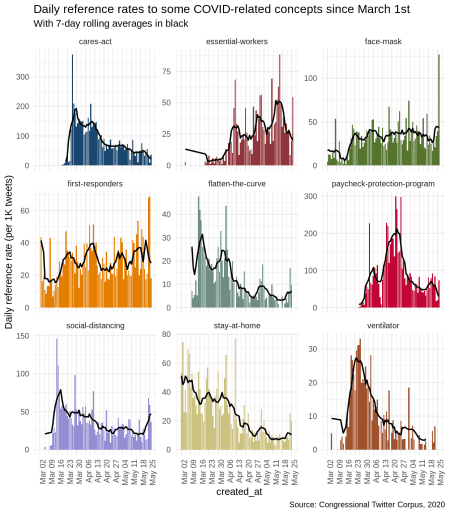
ylab('Daily reference rate (per 1K tweets)') +

facet\_wrap(~descriptor\_name, scales = 'free\_y') +

labs(title = 'Daily reference rates to some COVID-related concepts since March 1st',

subtitle = 'With 7-day rolling averages in black',

caption = 'Source: Congressional Twitter Corpus, 2020')



**§ Party prevalences**

Next, we consider variation in reference to COVID19 concepts among US lawmakers as a function of party affiliation. For COVID19 concept **X**, then, prevalence is calculated as the quotient of (1) the rate of reference for concept **X** among Republicans and (2) the rate of reference for concept **X** among Democrats. We tweak this quotient some such that the number reflects a percentage increase.

For a similar analysis on how party affiliation influences how US House Reps refer to the President of the USA on Twitter,

descriptor\_by\_party <- dtm3 %>%

filter(party != 'Independent') %>%

group\_by(party) %>%

mutate(party\_total = length(unique(doc\_id))) %>%

group\_by(party, descriptor\_name, party\_total) %>%

summarize(n = n()) %>%

group\_by(descriptor\_name) %>%

mutate(descriptor\_total = sum(n)) %>%

ungroup() %>%

filter(descriptor\_total > 40) %>%

mutate(per\_1k = round(n/party\_total \* 1000, 2)) %>%

select(-n, -party\_total) %>%

spread(party, per\_1k) %>%

mutate(ratio = ifelse (Democrat < Republican,

(Republican/Democrat) - 1,

(-Democrat/Republican) +1 )) %>%

mutate(ratio = round(ratio \* 100, 2)) %>%

filter(![is.na](http://is.na)(ratio) & abs(ratio) < 200)

The table below illustrates several examples of the relative prevalences of COVID19-related concepts as a function of political affiliation. Interpreting table, eg: Republicans are ~30% more likely than Democrats to reference american-dream on Twitter.

| **descriptor\_name** | **descriptor\_total** | **Democrat** | **Republican** | **ratio** |
| --- | --- | --- | --- | --- |
| american-dream | 112 | 2.07 | 2.71 | 30.92 |
| cares-act | 3476 | 62.82 | 86.22 | 37.25 |
| civil-liberty | 75 | 1.30 | 1.95 | 50.00 |
| close-contact | 155 | 4.04 | 1.84 | -119.57 |
| community-spread | 54 | 1.37 | 0.70 | -95.71 |
| coronavirus | 16397 | 313.64 | 378.67 | 20.73 |

**COVID19-concepts politicized**. A Democrat-Republican continuum. Concepts in green are ~neutral. Certainly some interesting differences, that shed light on both ideology and constituency demographics.

descriptor\_by\_party %>%

mutate(col1 = ifelse(ratio > 0, 'red', 'blue')) %>%

mutate(col1 = ifelse(ratio > -25 & ratio < 25, 'x', col1)) %>%

ggplot(aes(x=reorder(descriptor\_name,

ratio),

y=ratio,

label=descriptor\_name,

color = col1)) +

geom\_hline(yintercept = 25,

linetype = 2, color = 'gray') +

geom\_hline(yintercept = -25,

linetype = 2, color = 'gray') +

geom\_point(size= 1.5,

color = 'darkgray') +

geom\_text(size=4,

hjust = 0,

nudge\_y = 5)+

annotate('text' , y = -75, x = 20, label = 'DEMOCRAT') +

annotate('text' , y = 100, x = 10, label = 'REPUBLICAN') +

ggthemes::scale\_color\_stata() +

theme\_minimal() +

labs(title="COVID19-related concepts",

subtitle = 'Prevalence by party affiliation') + ##?

theme(legend.position = "none",

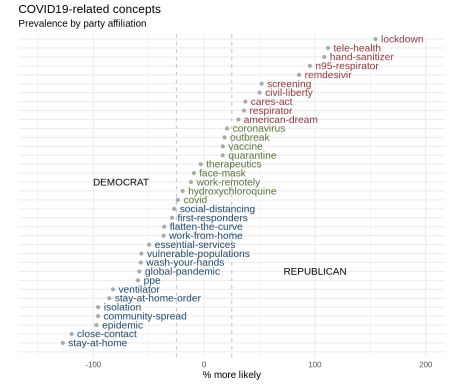
axis.text.y=element\_blank(),

axis.ticks.y=element\_blank())+

xlab('') + ylab('% more likely')+

ylim(-150, 200) +

coord\_flip()



**GloVe model & COVID19 semantic space**

The next piece is to build a GloVe model to investigate semantic relatedness among concepts included in our COVID19 lexicon. The general workflow here is:

1. Build a term-co-occurrence matrix (TCM),
2. Build an *n*-dimensional GloVe model based on the TCM,
3. Further reduce GloVe dimensions via tSNE, PCA, or MDS,
4. Plot terms in a reduced 2D space.

Here, we have the additional task of **aggregating the TCM from terms to descriptors** (or concepts), before building the GloVe model. The code below creates a simple table that crosswalks terms to concepts.

term\_vocab3 <- term\_vocab2 %>%

rename(term\_name = term) %>%

left\_join(dictionary) %>%

mutate(descriptor\_name = ifelse([is.na](http://is.na)(descriptor\_name),

term\_name,

descriptor\_name),

category = ifelse([is.na](http://is.na)(category),

'other',

category)) %>%

arrange(term\_name)

**§ Term-co-occurrence matrix**

Utilizing previously constructed text2vec primitives, we use the text2vec::create\_tcm function to construct a term-co-occurrence matrix, specifying a context window-size of 5 x 5.

tcm <- text2vec::create\_tcm(it = it\_phrases,

vectorizer = vectorizer,

skip\_grams\_window = 5L)

Then we implement the lvdr\_aggregate\_matrix function from the lexvarsdatr package to aggregate term vectors to a single descriptor vector (for forms included in the COVID19 lexicon).

tcm <- tcm[, order(colnames(tcm))]

tcm <- tcm[order(rownames(tcm)), ]

tcm1 <- lexvarsdatr::lvdr\_aggregate\_matrix(tfm = tcm,

group = term\_vocab3$descriptor\_name,

fun = 'sum')

Dimensions of TCM:

## [1] 42468 42468

**§ GloVe model**

We specify GloVe model parameters via the text2vec::GlobalVectors function, and build term vectors using fit\_transform. Vectors are comprised of n = 128 dimensions.

set.seed(99)

glove <- text2vec::GlobalVectors$new(rank = 128,

#vocabulary = row.names(tcm1),

x\_max = 10)

wv\_main <- glove$fit\_transform(tcm1,

n\_iter = 10,

convergence\_tol = 0.01)

wv\_context <- glove$components

glove\_vectors <- wv\_main + t(wv\_context)

**§ Semantic & conceptual relatedness**

With GloVe vectors in tow, options abound. Here, we demonstrate two fairly straightforward applications. The first – a quick look at **nearest neighbors** for a set of COVID19-related concepts. Via cosine similarity and the LSAfun::neighbors function.

eg\_terms <- c('stay-at-home', 'outbreak',

'front-line-workers', 'vaccine',

'relief' )

x <- lapply(eg\_terms,

LSAfun::neighbors,

glove\_vectors,

n = 10)

names(x) <- eg\_terms

**Top 10 nearest neighbors** for stay-at-home, outbreak, front-line-workers, vaccine & relief. So, despite a relatively small corpus, some fairly nice results.

## $`stay-at-home`

## stay-at-home practice social-distancing flatten-the-curve

## 1.0000000 0.5634366 0.5500531 0.4852880

## sick stay avoid staying

## 0.4788068 0.4424411 0.4312511 0.4096375

## unless you

## 0.3959533 0.3679505

##

## $outbreak

## outbreak coronavirus pandemic covid covidー19 crisis

## 1.0000000 0.6616114 0.5697251 0.5494554 0.5296838 0.4987309

## response spread virus impacts

## 0.4826594 0.4798837 0.4642697 0.4621730

##

## $`front-line-workers`

## front-line-workers essential-workers first-responders ppe

## 1.0000000 0.5883052 0.5341892 0.4832370

## heroes professionals nurses frontline

## 0.4822017 0.4641108 0.4569190 0.4568646

## doctors equipment

## 0.4491088 0.4448107

##

## $vaccine

## vaccine treatments develop therapies treatment development

## 1.0000000 0.5817043 0.4566783 0.4113966 0.4112393 0.3854895

## easy tests genesis research

## 0.3832985 0.3499214 0.3297194 0.3265058

##

## $relief

## relief funding aid additional provide bill support

## 1.0000000 0.6063464 0.5879983 0.5798526 0.5768965 0.5732163 0.5670530

## package assistance cares-act

## 0.5640401 0.5563580 0.5379026

For a smarter approach to visualizing nearest neighbors, as well as visualizing lexical semantic change historically,

In the second application, we consider a **two-dimensional perspective on the semantic relatedness** of concepts included in our COVID19 lexicon. While we have built vectors for all forms attested in the Congressional Twitter Corpus, here we subset this matrix to just COVID concepts. Via classical multidimensional scaling, we project 128 features (per GloVe) into two-dimensional Euclidean space.

set.seed(99)

keeps <- descriptor\_freq %>% filter(term\_freq > 30)

glove1 <- glove\_vectors[rownames(glove\_vectors) %in%

unique(keeps$descriptor\_name),]

sim\_mat <- text2vec::sim2(glove1,

method = "cosine",

norm = "l2")

# data set too small for tSNE ---

y1 <- cmdscale(1-sim\_mat, eig = TRUE, k = 2)$points %>%

data.frame() %>%

mutate (descriptor\_name = rownames(sim\_mat)) %>%

left\_join(dictionary %>% distinct(descriptor\_name, category))

**A semantic map** of COVID19 concepts is presented below. Some intuitive structure for sure; some less so.

y1 %>%

ggplot(aes(X1,X2, label = descriptor\_name)) +

geom\_point(aes(color = category), size = 3.5) +

ggrepel::geom\_text\_repel(

data = y1,

nudge\_y = 0.025,

segment.color = "grey80",

direction = "y",

hjust = 0,

size = 3 ) +

ggthemes::scale\_colour\_stata() +

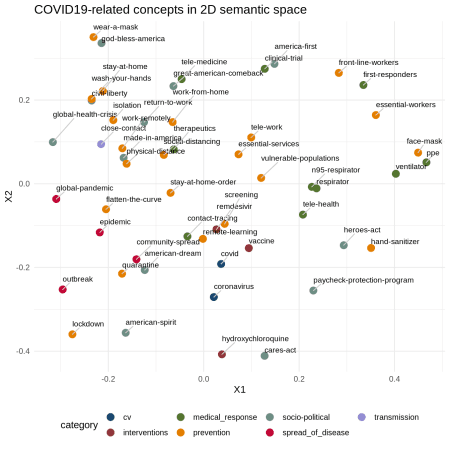
theme\_minimal() +

#theme\_classic() +

theme(legend.position = "bottom",

plot.title = element\_text(size=14))+

labs(title="COVID19-related concepts in 2D semantic space")



**Networks & lexical co-occurrence**

Lastly, we build & visualize a co-occurrence network based on the previously constructed term-co-occurrence matrix. The lexvarsdatr package streamlines these processes, and enables straightforward extraction of sub-networks from large matrices.

Below, we convert our count-based TCM to a *positive point-wise mutual information*-based matrix (via lvdr\_calc\_ppmi), and extract the 20 strongest collocates (via lvdr\_extract\_network) for five (cherry-picked) concepts included in the COVID19 lexicon.

network <- tcm1 %>%

lexvarsdatr::lvdr\_calc\_ppmi(make\_symmetric = TRUE) %>%

lexvarsdatr::lvdr\_extract\_network (

target = c('contact-tracing',

'flatten-the-curve',

'return-to-work',

'social-distancing',

#'remote-learning',

'drive-through-testing'),

n = 20)

**And then visualize**:

set.seed(66)

network %>%

tidygraph::as\_tbl\_graph() %>%

ggraph::ggraph() +

ggraph::geom\_edge\_link(color = 'darkgray') +

ggraph::geom\_node\_point(aes(size = value,

color = term,

shape = group)) +

ggraph::geom\_node\_text(aes(label = toupper(label),

filter = group == 'term'),

repel = TRUE, size = 4) +

ggraph::geom\_node\_text(aes(label = tolower(label),

filter = group == 'feature'),

repel = TRUE, size = 3) +

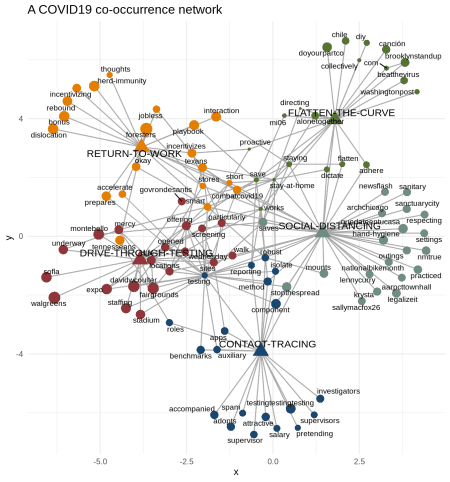
ggthemes::scale\_color\_stata()+

theme\_minimal() +

ggtitle('A COVID19 co-occurrence network') +

theme(legend.position = "none",

plot.title = element\_text(size=14))



**Summary**

So, more of a resource/guide than a post-proper. Mostly an attempt on my part to collate some scattered methods. And a bit of an ode totext2vec.