COVID-19に対する「社会」の声を聞く：Rの学習(3)

資　料

COVID-19に対する「社会」の声を聞く – Rの学習(3)

遠　山　弘　徳

Ⅰ.「社会」の声

ときには暴力的にもなり，悲惨な結果をもたらすこともありますが，ソーシャルメディア(以下，SNS)は今では１人ひとりが自分の「声」を社会に発信するための人気のメディアとなっています．SNSは，良い意味でも悪い意味でも，マスメディアのフィルターを通さないため，よりストレートに人々の声を伝えることができます．

SNSのデータは一般的に，個々のユーザーが作成し，公開プラットフォームを使用して収集されます．これらの公開プラットフォームには，Twitter，Google，Facebook，InstagramなどのSNSが含まれます．こうして収集されたSNSのデータは，経済，政治などの社会の出来事や自然災害などについて，ほぼリアルタイムで発信される人々の考え，言いかえれば今現在の「社会」の声を知る上で有益な情報ソースとなっています．

今回はSNSを通して新型コロナウィルスに対する「社会」の声を聞く方法をとりあげます．人々がどの程度新型コロナウィルスに関心を持っているのか，また新型コロナウィルスはどのように受け止められているのか等，こうした点を知るために，SNSで発せられたテキストをRを使って分析します．これにより，新型コロナウィルスに関する「社会」の声の一端を理解できると思います．

本号では，Rによるテキスト・データの処理・分析が学習の中心となります．テキストデータの分析には，tidyverseの設計思想にもとづいて作成されたRパッケージtidytextが有益な分析ツールとなります．

Ⅱ.「社会」の関心を知る

人々は新型コロナウィルスにどれほど関心を寄せているのでしょうか．これを知るために，さまざまな機関が行うアンケート調査からマスメディアが実施する大規模な世論調査にまで及ぶ，さまざまな調査が利用可能です．こうした調査は有益な情報を提供し，自治体・政府の政策立案の基礎となることもあります．しかし，こうした調査は信頼性は高いものの，リアルタイムで人々

の関心を知るにはどうしても遅れますし，「生」の声 ― 調査機関，マス・メディアのフィルターを通さない声―を知るには限界があります．

わたしたちは常日頃，何かに対する情報を知りたいとき，インターネット検索を行っています．その意味では検索はわたしたちの関心を示していると言っても良いでしょう．インターネット検索の巨人は，言うまでもなく，Googleです．Googleは膨大な検索データを蓄積していますが，これをもとにGoogleトレンド(https://trends.google.com/trends/?geo=JP)と呼ばれるサービスを提供しています．Googleトレンドは人々の関心を知る上で強力なツール[1]です．トレンドの計算にあたっては検索のピーク時を100としたときの相対指数です．したがって数値の大きさは検索条件を変更すると変わってしまうことに留意する必要があります．

Googleトレンドでは，地域，期間を指定し，任意の検索キーワードの相対的な人気の程度の推移を調べることができます．Rには，こうしたGoogleトレンドのデータを取得するパッケージgtrendsRやtrendyが開発されています．

本資料ではgtrendsRを利用します．それではこのパッケージをインストールし，新型コロナウィルスに対する人々の関心を見ることにしましょう．インストールするためにはコンソール画面に次のように入力し，エンターキーを押してください．

```
install.packages(“gtrendsR”)
```

次にこのパッケージを利用するために，スクリプト画面に

```
library(gtrendsR)
```

と入力し，実行[Run]をクリックします．これでgtrendsRを利用できます．最初に，Googleトレンドページでも調べることができますが，特定のキーワードを検索し，その推移を見てみましょう．このためのgtrendsRの基本的なコードは次のようになります．

```
gtrends(keyword=“ ”, geo =“ ”, time =“ ”)
```

- keyword=“ ”には検索キーワードを入力します．複数のキーワードを入力したい場合はc( )を使い，たとえば，keywords = c(“新型コロナウィルス”, “抗原検査”)と入力します．

[1] たとえば，Askoy, et al. (2020)は，COVID-19に関連した１国の１日のGoogle検索のシェアを国民の注目度とし，注目度が非医学的な介入政策―たとえば，ロックダウン政策―とどのように関連しているかを検討している．

- geo=“ ”ではiso2コードを使って検索地域を指定します．デフォルトではall，つまり全世界です．複数の地域を指定したい場合は，同じくc( )を使います．たとえば，日本，アメリカ，中国を指定したい場合，geo=c(“JP”, “US”, “CN”)と入力します．
- time=“ ”によって検索期間を指定できます．たとえば，
  過去４時間の場合はtime = “now 4 -H”
  過去７日間の場合はtime = “now 7 -d”
  過去３ヶ月の場合はtime = “today 3 -m”
  過去５年間の場合はtime = “today + 5 -y”
  任意の期間指定の場合，“Y-m-d Y-m-d”を使います．たとえば，2019年６月27日から2020年１月21日までを指定したい場合，time = “2019-06-27 2020-01-21”と入力します．

それではgtrendsRを使って，キーワードを「COVID-19」,「PCR」,「AKB」，地域を「日本」，期間を「2020年１月21日から2020年６月27日」とし，検索トレンドをみてみましょう．「AKB」は比較のために入れていますが，国民的アイドルを凌ぐ「COVID-19」と「PCR」の「相対的な人気」が分かります．スクリプト画面に次のように入力し，実行[Run]してください．

```
trend <- gtrends(keyword=c(“COVID-19”,“PCR”,“AKB”), geo=“JP”, time = “2020-01-21 2020-06-27”)
```

この例では結果をtrendという名前をつけたオブジェクトに容れています．

```
summary(trend)
```

を実行すると，次のオブジェクトリストが表示されます．

	Length	Class	Mode
interest_over_time	7	data.frame	list
interest_by_country	0	-none-	NULL
interest_by_region	5	data.frame	list
interest_by_dma	0	-none-	NULL
interest_by_city	5	data.frame	list
related_topics	0	-none-	NULL
related_queries	6	data.frame	list

summary()で確認されたtrendの中には上記のようなデータフレームが含まれています．

- interest_over_timeは時間を通じた関心であり，検索キーワードの期間のヒット数を示すデータフレームです．
- interest_by_regionは地域ごとの関心を示すデータフレームです．
- interest_by_cityは同様に都市ごとの関心を示すデータフレームです．
- related_queriesは関連した検索語が入ったデータフレームです．

この他，国ごとのデータ，DMAごとのデータ，関連したトピックごとのデータもありますが，ここでの例では国をJP，複数の検索ワードを指定したために，いずれもNULL(データなし)となっています．オブジェクトtrendの中のinterest_over_time(時間を通じた関心)はgtrendsRの中のplot.gtrends関数によって簡単にグラフ化できます．次のように入力してみてください．図1のようなグラフが表示されます

```
plot(trend)
```

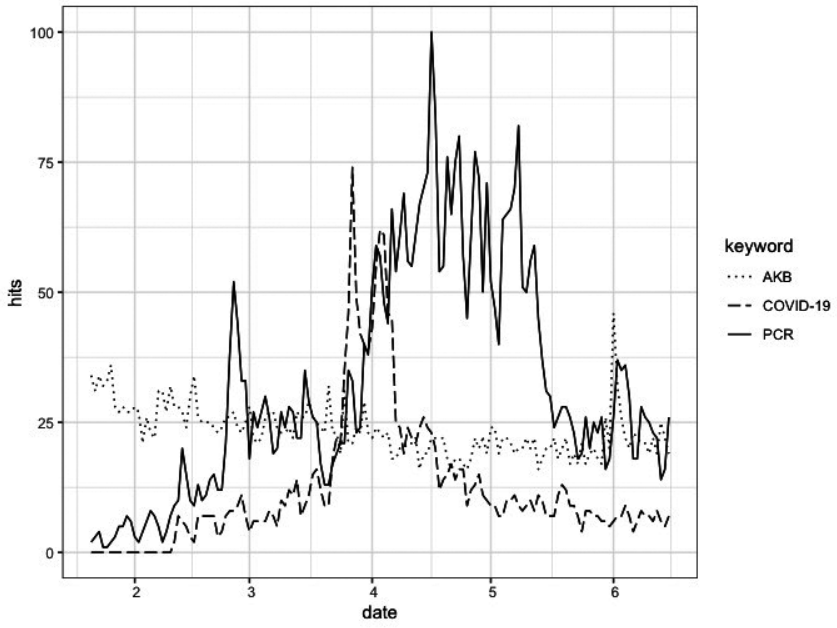

図1　COVID-19, PCR, AKBの検索推移

注．plot()によってカラーで区別された線が表示されるが，ここではggplot2を利用し，白黒グラフを作成．

図１をみると，２月半ばまでは人々の関心は，新型コロナウィルスよりも，国民的人気アイドルAKBにあったことが分かります．しかし，３月末から４月初頭にCOVID-19に対する関心は急上昇します．PCR検査はいったん２月末に大きな関心を呼びます．これはおそらくダイヤモンドプリンセス号の新型コロナウィルスの集団発生に起因すると思われます．その後，PCRに対するヒット数も低下しますが，しかし感染者数が拡大するにつれて，４月にふたたび大きく注目されています．COVID-19，PCR検査の「人気度」は５月頃から落ち着きを見せ始め，６月の終盤からは再びAKBの「相対的人気」と同等か，それを下回ります．AKBの人気が大幅に伸びていくことを期待したいものです．

Ⅲ．「社会」の声を集める

ツイッターをつうじて，新型コロナウィルスに関する社会の声を集めることにします．Rにはツイッターからデータを収集する便利なパッケージrtweetが開発されています．他にもtwitteRというパッケージもありますが，今ではrtweetがツイッター・データにアクセスするための標準的ツールとなりつつあります．

ツイッター情報を取得するためには，API(Application Programming Interface)認証という面倒な事前準備が必要でしたが，rtweetを利用することで，必要なのはTwitterアカウント(ユーザー名とパスワード)だけとなりました．認証を受けるには，Rの対話型セッション中にTwitterのAPIにリクエストを送るだけです．つまり，search_tweets()，get_timeline()などのrtweetパッケージの関数を使用するだけです．自分のTwitterアカウントに代わってWebブラウザのポップアップから認証を行うことができます．これによりトークン(パスワード生成)が作成され，今後の利用のために保存されます．

ツイッターデータ収集のために，rtweetは多くの関数を提供しています．たとえば，現在の日本のトレンドを知りたい場合は次のように入力します．

```
get_trends(“japan”)
```

収集されたトレンドワードをみてみましょう．

```
get_trend(“japan”) %>%
        select(trend) %>%          # select()関数で変数trendの列だけを取り出す
              View()               # View()関数で表示
```

	trend
1	レジ袋
2	ディズニー
3	#ローソンの生ブッセ
4	NAVER
5	#カラオケマック
6	#TDR_now
7	攻撃II
8	やべっちFC
9	N700S
10	ソアリン
11	#ルムマで毎日1000パック
12	#絶対カナダうまいでしょ
13	下半期
14	マイバッグ
15	雲メイク
16	ラウちゃん
17	攻撃2
18	半夏生
19	日銀短観
20	トイマニ

図2　収集されたツイッターのトレンドワード
注．2020年7月1日に取得．

7月1日時点のトレンドワードですので，ツイッター上のつぶやきは「レジ袋」の有料化や「ディズニー」の再開についてのものが多いようです．経済関連では「日銀短観」がトレンドワードに入っています．

rtweetパッケージの中でもっとも有益な関数はsearch_tweets()です．search_tweets()は検索語queryによって指定されたツイートデータを得ることができます．基本的なコードは以下のようになります，

```
seach_tweets(q="検索語", n=取得するツイッターの数)
```

- q= "corona virus"と入力すると，"corona"と"virus"の両方の検索語を含んだツイートを探します．つまり検索語の間にスペースを容れると，and検索になります．OR検索を利用する場合は，q="corona OR virus"とします．

- 取得できるツィート数にはAPIの制限があります．ツィーターの場合は，６～９日分の過去のツイート，１回のAPIコールで18, 000ツイート，１時間に100回のリクエストに制限されています．このため特定のイベント(例えば自然災害や選挙やなど)についてデータを集めるさいには，先を考えてデータ収集を行う必要があります．

それでは新型コロナウィルスに関するツイートを集めてみましょう．新型コロナウィルスに関連した検索後としては“covid-19”，“coronavirus”が考えられます．両方のいずれかを含む(つまりOR検索)ツイートを探してみましょう．上限を18, 000ツイートとしておきます．さらに，リツイートは除外するために，include_rts=FALSE(リツイートを含める場合はTRUEです)を入れ，言語を英語に指定するために，引数lang=“en”と指定します．そして結果をオブジェクトtweet_covidに容れます．

```
tweet_covid <- search_tweets(q=“covid-19 OR coronavirus”,n=18000,include_rts = FALSE,lang=“en”)
```

これによってtweet_covidというデータフレームが出来上がります．コンソール画面に“tweet_covid”と入力すれば，内容が表示されます．そのそれぞれの行(観察値)は異なったツイートです．過去９日間(実際には，数時間のうちに18, 000のツィートに達してしまいます)に，このキーワードを含むツィートがどれだけつぶやかれたのかをグラフにしてみましょう

rtweetパッケージの中のts_plot()関数によってその頻度を簡単にみることができます．次のように入力すると図３が描かれます．

```
ts_plot(tweet_covid，by=“mins”)
```

ts_plot()関数のもっともシンプルな用法は２つの引数を指定するだけです．

```
ts_plot(データフレーム名，by=“時間間隔の指定”)
```

データフレーム名はこの例ではtweet_covidです．by=によって時間の間隔“mins”，“hours”，“days”，“weeks”を指定できます．たとえば３日刻みを指定する場合は，by = “3 days”とします．この例ではby = “mins”によって１分の間隔を指定しています．

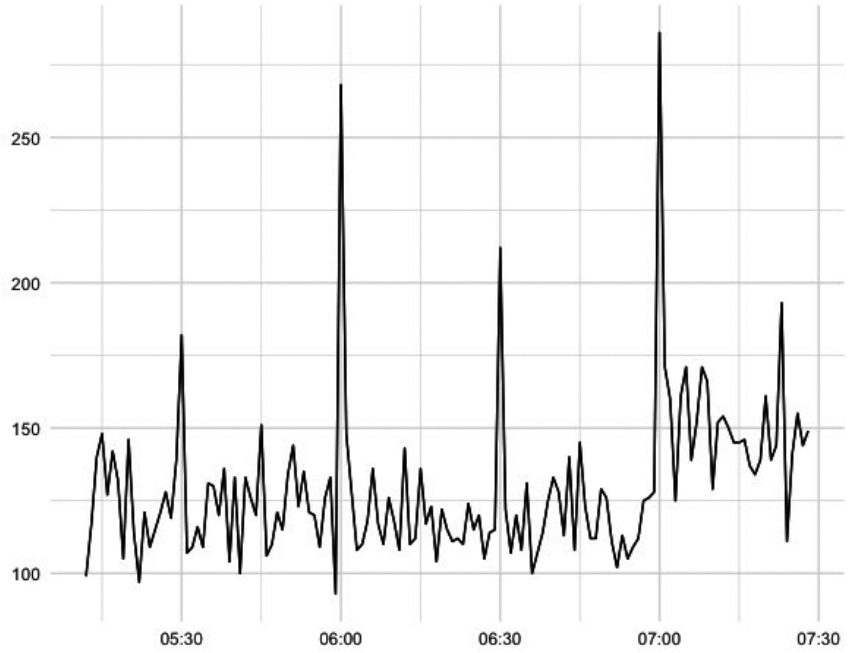

図３　COVID-19もしくはcoronavirusを含んだツィート数の推移
注．ツイッターデータの取得は2020年６月27日16時40分(日本時間)．
またggplot2のtheme_bw()を利用し，背景を白黒に変更してある．

Ⅳ．「社会」の声を聞く―ツィッターテキストの分析

それでは「社会」の声を聞いてみましょう．ツイッターのテキストデータの分析には，Silge，Robinsonの両氏によって開発されたtidytextパッケージを利用することになります．

最初に，rtweetで集めたテキストデータを抜き出し，テキスト分析に利用しやすいようにデータを編集する必要があります．その上で，テキストデータを視覚的に表現することにします．おもな学習内容は次のとおりです．

１．文書の分割 ― トークン化
２．nグラムへのトークン化
３．可視化 ― ワードクラウドとネットワーク
４．センチメント分析

ここで言うトークンとは，テキストの単位として意味のあるもの ―たとえば，単語―であり，テキスト分析の対象となるものです．したがってトークン化とはテキストをトークンに分割することです．

Ⅳ－１　文書の分割とトークン化

それでは具体的な作業に入ることにしましょう．おもな作業と利用関数は次のとおりです．

⑴　文書を単語に分割　　unnest_tokens()
⑵　ストップワーズの除去　　anti_join(stop_words)
⑶　頻出語の可視化　　wordcloud

分析の対象はツイッターによってつぶやかれたテキスト部分です．そこで最初に，データフレームtweet_covidからつぶやかれた時間created_atとテキストtextの２つの変数を取り出しておきましょう．そしてその結果をオブジェクトtext_twに容れます．

```
text_tw <- tweet_covid %>%
        select(created_at,text)      # select()を使って特定の変数列を抽出
```

View(text_tw)によってテキスト部分―text変数―をみると，たとえば，

> “@West_GP @mikegalsworthy You’re all missing the point here. The UK is THE Covid-19 test site.”

のように長い文書(複数の単語を連結したもの)になっています．これではテキスト分析には適しません．テキスト分析のためには，さらに，それぞれのツィート文書を単語wordに分割する必要があります．テキスト文書を，整理された形式(tidy形式)で単語に分割するために，Rパッケージtidytextのunnest_tokens()関数を使います．

この関数を利用するために，パッケージtidytextをインストールし，library()で呼び出しておきましょう．

```
install.packages(“tidytext”)
library(tidytext)
```

それぞれ実行[Run]しておいてください．unnest_tokens()関数の書き方は次のようになります，

```
unnest_tokens(データフレーム名，作成される変数列名，単語に分割する変数列名)
```

上述のスクリプトにパイプでunnest_tokens()の命令を付け加えることにします．この例ではパイプ(％＞％)でつなげていますので，unnest_tokens()関数の中にデータフレーム名を書く必要はありません．

```
text_tw <- tweet_covid %>%
select(created_at,text) %>%
unnest_tokens(word,text)       # unnest_tokens()を使ってtextをwordに分割
```

この結果をView(text_tw)でデータフレームの中をみると，図４のように，textが単語に分割され，word変数が作成されたことが分かります．

	created_at	word
1	2020-06-27 07:28:58	west_gp
2	2020-06-27 07:28:58	mikegalsworthy
3	2020-06-27 07:28:58	you're
4	2020-06-27 07:28:58	all
5	2020-06-27 07:28:58	missing
6	2020-06-27 07:28:58	the
7	2020-06-27 07:28:58	point
8	2020-06-27 07:28:58	here
9	2020-06-27 07:28:58	the
10	2020-06-27 07:28:58	uk
11	2020-06-27 07:28:58	is
12	2020-06-27 07:28:58	the
13	2020-06-27 07:28:58	covid
14	2020-06-27 07:28:58	19
15	2020-06-27 07:28:58	test
16	2020-06-27 07:28:58	site
17	2020-06-27 07:28:57	pakistan
18	2020-06-27 07:28:57	was
19	2020-06-27 07:28:57	origin
20	2020-06-27 07:28:57	for
21	2020-06-27 07:28:57	half
22	2020-06-27 07:28:57	of

図４　データフレームtext_tw
注．473,174行と２列からなる．

⑵　ストップワーズの除去 ― anti_join(stop_words)

これによって図４のような整理されたデータフレーム(tidy形式)ができあがります．しかし，この図４の変数wordの列をみると，the，for，ofなどの，よく使われますが，重要ではない単語も抽出されています．こうした単語はストップワーズstop wordsと呼ばれます．テキスト分析にはストップワーズは不要ですので取り除くとします．

tidytextの中にはストップワーズを集めたデータセットstop_wordsが事前に用意されています．これとanti_join()関数を使ってstop wordsを取り除きます．anti_join()関数は基本的に２つのデータフレームを結合する関数ですが，anti_から想像されるようにマッチしなかった行からなるデータフレームを作成します．書式は以下のとおりです．

```
anti_join(x,y)
```

このコードは，データフレームxの行が，データフレームyの行とマッチしなかったすべての行を返します．

```
data(stop_words)                    # ストップワーズのデータセットを読み込みます
```

```
text_tw <- text_tw %>%
        anti_join(sop_words)        # anti_join()を使ってstop_wordsとマッチしない行を取得
```

この例でもパイプを使用していますので，最初のデータフレーム名は省略されます．anti_join()によってデータフレームstop_wordsの中に入っている行と一致しない，データフレームtext_twの行を返すことになります．つまり，text_twの中にあったofやtoといったデータを持つ行がすべて取り除かれます．

しかし，text_twの中をみると，さらに，不要なword―おもにサイト名等を表現する単語―が抽出されているようです．これはfilter()関数を使って除去しておきます．最初に，不要なwordをピックアップし，まとめてno_wordsに容れます．

```
no_words <- c(“t.co”, “https”, “it’s”, “1”, “2”, “20yearshuaweieurope”,
        “tech4all”, “ixqblwihjg”, “__ice9”, “sir”,“protocol”, “karachistockexchange”, 以下省略)
```

```
text_tw <- text_tw %>%
        filter(!text %in% no_words)                # filter()関数を使って除去.
```

filter(条件)は条件にあった行を返しますが，ここでは条件の前に“!”をつけていますので，条件にあった行以外の行を返します．つまり，不要なword以外のwordを持つ行を返します．また，“x %in% y”は論理和を表現し，「xがyの中のどれかに一致する」場合を意味します．この例ではtextの中のwordが，no_wordsの中のどれかのwordに一致する，ということを意味します．

これで分析に利用可能な，98行×２列のデータフレームtext_twが作成されました．次に，新型コロナウィルスに関わるツィッターにおいて「何」が―正確にはどのような単語が注目されているのかをみるかをために，wordcloudという手法を適用します．

⑶　頻出語の可視化― wordcloud

wordcloudとは，文章中で出現頻度の多い単語を選び出し，頻出語をその頻度に応じた大きさで雲のように描く手法です．この例で言えば，取得したツイッター文書からもっとも多く見られる単語を可視化することになります．wordcloud[2]を使ってツイッター文書text_twの中の頻出語を可視化した結果が図５です．

wordcloud()関数を使って頻出語を出力するためには，tmというRパッケージも必要となります．そこで事前準備としてインストールし，呼び出しておきます．それぞれ次のように入力し，実行[Run]してください．

```
install.package(“tm”)          #パッケージtmのインストール
library(tm)                    # tmの呼び出し
```

wordcloud()関数の基本コードは次のようになります，

```
wordcloud(出力させる単語の列，出現頻度数，max.words=図示する最大単語数)
```

それではこの書式にしたがってツイッター文書text_twにwordcloud()を適用してみましょう．

[2] wordcloudを描く関数としては他にwordcloud2，geom_wordcloudがあります．

```
wordcloud(text_tw$word,text_tw$n, max.words = 100)
```

ここではデータフレーム名を明示する必要があります．つまりどのデータフレームの中のどの列を利用するかをRに教える必要があります．データフレーム名を明示するために，text_tw$を使っています．また，表示する単語数はmax.words=によって100単語としています．これを実行すると，以下の図５が得られます．

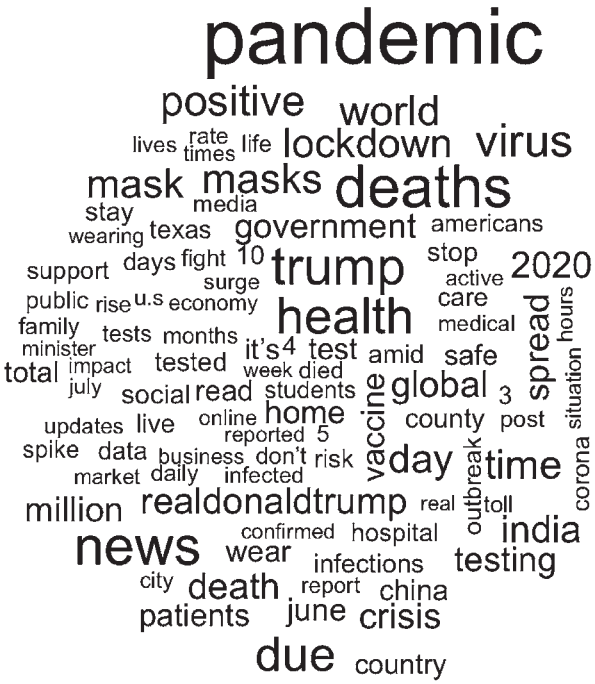

図５　ツイッターテキストにおいて新型コロナウィルスとともにもっとも頻繁に利用される単語
注．ツィートデータは６月30日取得．

人々がツイッターを使って新型コロナウィルス について発信するとき，おもにどのような単語が利用されているか．このwordcloudによって視覚的に捉えることができます．圧倒的にpandemicです．次いで，deaths，health，trumpといった単語，さらに，positive，masks，lockdown，newsといった単語の出現頻度が高いということが理解されます．人々が新型コロナウィルスについてはパンデミック，死者，健康，陽性反応positiveに強い関心を寄せていることが分かります．また，新型コロナウィルスについてつぶやくとき，トランプアメリカ大統領とロックダウンlockdown，maskに言及することも多くあるということが分かります．これは同政権に対する公衆衛生政策に関連しているのかもしれません．

じっさい，この時期，新型コロナウィルスの拡大が収まらず，世界全体で確認された感染者が1,020万人，死者が10万人を超えたと発表された時期でした．とくにアメリカではアリゾナ，フロリダ州等で感染者数がふたたび急増し，第２波への恐怖が人々を捉えていた時期でした．

ここで利用されたデータは１時点および世界中のツィートデータですが，時間経過を追跡したり，地域別のツィートデータを比較したりすれば，より興味深い結果が得られるかもしれません．

図６は，イギリス・ロンドンでのツィートのwordcloudです．rtweetパッケージでは特定の地域を指定し，ツィート情報を取得することもできます．そのためにはsearch_tweet()関数の引数geocode=に緯度・経度・半径の情報を指定します．たとえば，ロンドンのツィートに限定したい場合，引数geocode=にロンドンの座標―緯度51.509865，経度-0.118092および半径100マイル―を入力します．

```
search_tweets(q="covid-19 OR coronavirus", n=18000,lang="en",
        include_rts = F,geocode = "51.509865,-0.118092,100mi")
```

これを利用することで，ロンドンに場所を限定し，同市を中心に半径100マイル以内で発信されたツイッターを収集できます．図６はこれによって得たツイッター情報にwordcloudを適用したものです．

図６　ロンドンにおけるツィート

注．ツイッターデータは７月４日17：30(日本時間)に取得．

図５では中心的な話題の１つはトランプ大統領でしたが，図６をみると，ロンドンの人々が新型コロナウィルスについてつぶやくとき，大きな関心はロックダウンlockdownにあるようです．政府governmentや国民保健サービスnhsという言葉も多くつぶやかれていることをみると，個人よりも政策が注目されているのかもしれません．

Ⅳ－２　nグラムとネットワークの可視化

nグラムとは任意の文字列・文書を連続したn個の単語に分割するテキスト分割法です．たとえば，nが２のときはバイグラム(bi-gram)，３個の場合はトライグラム(tri-gram)と呼ばれます．これまでは独立した単位としての単語 ― n = 1のユニグラム ― に注目してきました．しかし，ある単語がある単語の後ろによく現れたり，同じ文書の中に同時に出現する単語もあります．こうした結びつき―あるいは共起表現―は，単語そのものよりも，分析対象のテキストについてより多くの情報を伝えてくれます．

そこで次に，隣り合う単語の関係つまりバイグラムに注目します．基本的な作業は同じです．違いは，文書テキストを１つの単語に分割するのではなく，「隣り合う単語バイグラム」への分割ということになります．１単語への分割と同様に，unnest_tokens()関数を使いますが，引数にtoken =“ngrams”を指定し，隣り合う単語の数を “n =” で指定します．基本的なコードは以下の通りです．

```
unnest_tokens(データフレーム名，作成される変数列名，分割するテキストの変数列名，
        token = “ngrams”,n=2)
```

この関数をツィートテキストに適用します．

```
unnest_tokens(tweet_covid,bigram,text,token = “ngrams”,n=2)
```

容易に予想されるように，これを実行すると，ただ単に隣り合う単語を抽出することになり，of，to，inなどのような前置詞と結合した２つの単語が引き出されるケースが多くなります．ペアの単語はbigramと名づけられた変数列に入っていますので，head()，View()で確認してみてください．

そこでbigram変数の中のペアの単語を，tidyrのseparate()関数で２つの列に分割し，そのいずれかがストップワーズならば分析対象からはずすことにします．separate()関数の基本的な書式は次のようになります．

```
separate(データフレーム名，分割される変数列名，作成される新変数列名)
```

それではここまで説明した作業をパイプでつなげて実行してみます．ダウンロードしたツイッターデータはtweet_covidに保存してあります．このテキスト変数にunnest_tokens()関数を適用し，さらに，separate()関数を適用し，twというオブジェクトに容れます．

```
tw <- tweet_covid %>%
        select(created_at,text) %>%
        unnest_tokens(bigram,text,token = “ngrams”,n=2) %>%
        separate(bigram, c(words1, words2))
```

これで変数word1，word2を持つtwオブジェクトができました．次に，オブジェクトtwの中の変数列word1，word2にfilter()関数を適用し，重要ではない単語のペアがある行を削除します．ここではtidytextの中のstop_wordsのデータとstop_words以外に重要ではないと思われる単語をピックアップし，そうした単語の入った行を削除します．最初に，重要ではない単語をピックアップし，no_wordsに容れます．

```
no_words <- c(“t.co”, “https”, “it’s”, “1”, “2”, “20yearshuaweieurope”,
        “tech4all”, “ixqblwihjg”, “__ice9”, “sir”, “protocol”,以下省略)
```

それではno_wordsの単語とstop_wordsの中の単語と一致する行を見つけ出し，それを削除するコードを作成します．

```
tw <- tw %>%
        filter(!word1 %in% no_words) %>%
        filter(!word2 %in% no_words) %>%
        filter(!word1 %in% stop_words$word) %>%
        filter(!word2 %in% stop_words$word)
```

これで不要な単語を除いた単語ペア―word1，word2―の入ったtwが作成されました．これを，単語ペアの出現頻度を数え，多い順に並び替え，オブジェクトtw_rankに容れます．

```
tw_rank <- tw %>%
        count(word1,word2, sort=TRUE)
```

これを実行し，tw_rankをView()で見ると，次の図７のように出現頻度順に並んだ単語のペアを抽出することができます．

Filter

	word1	word2	n
1	white	house	475
2	house	readies	309
3	tested	positive	264
4	jimmie	johnson	233
5	top	story	189
6	secret	service	187
7	test	positive	185
8	mount	rushmore	182
9	social	distancing	182
10	realdonaldtrump	flotus	170
11	whitehouse	realdonaldtrump	167
12	service	agents	159
13	tests	positive	157
14	death	rate	141
15	hydroxychloroquine	helped	141
16	mexico	closes	129
17	trump	campaign	118
18	8	secret	115
19	agents	stuck	109
20	patients	study	107

図７　出現頻度の多い単語ペア

注．ツイッターデータは７月４日午前10時(日本時間)に取得．データフレームは58, 823行であるが，この図では最初の20行のみ表示．

この図7を見ると，どの単語がどの単語と結びついているかを理解できます．たとえば，whiteとhouse，testedとpositive，socialとdistancingといったお馴染みの単語のペアを見ることができます．これ以外に，hydroxychloroquine(ヒドロキシクロロキン)とhelpedといった耳慣れない組み合わせが注目されていることも分かります．ヒドロキシクロロキンは抗マラリア剤ですが，それがhelpedと結びついていますので，新型コロナウィルスにヒドロキシクロロキン[3]が効果があるかどうかが人々の関心を集めているのかもしれません．

しかし，59,000近くのペアから何か意味ある関係を読み取ることは容易ではありません．そこでこれをグラフにし理解を容易にしたいと思います．つまり単語の結びつきをネットワークとして描きます．そのためには接続された節の点の組み合わせ―「辺の起点」「辺の終点」および「辺に与えられた属性」―が必要となります．データフレームの列を接続された節の組み合わせに変換するためには，igraphパッケージのgraph_from_data_frame()を利用します．そこで最初に，このRパッケージをインストールし，利用できるようにしておきましょう．

```
install.packages(“igraph”)
library(igraph)
```

それぞれ実行[Run]してください．これでgraph_from_data_frame()関数が利用できます．この関数の基本的なコードは次のようになります．

```
graph_from_data_frame(データフレーム名，directed= TRUE, vertices = NULL)
```

引数verticesは，頂点を持つデータフレームを指定しなければ，デフォルトでNULLです．NULLの場合，データフレームの最初の2つの列が順に「辺の起点」,「辺の終点」として利用されます．したがってもっとも単純なコードはgraph_from_data_frame()の中にデータフレームを記すだけです．ただし，データフレームの最初の2つの列は，上述のように，辺の起点，終点となり，追加の列は「辺に与えられた属性」となるため，少なくとも3列のデータが必要となります．引数directed=は有向か無向かを指定します．

```
tw_fig <- graph_from_data_frame(tw_rank)
```

[3] 米国立衛生研究所は6月20日，ヒドロキシクロロキンが新型コロナウィルスの入院患者に有益な効果をもたらす可能性は非常に低いと結論づけ，臨床実験の中止を発表しています．

これでネットワークが作成されます．この例では結果をオブジェクトtw_figに容れていますので，コンソール画面にtw_figと入力し，エンターキーを押すと，以下のような画面が表示されます．

```
> tw_fig
IGRAPH bcbdccf DN-- 144 100 --
+ attr: name (v/c), n (e/n)
+ edges from bcbdccf (vertex name:
 [1] white          ->house
 [2] house          ->readies
 [3] tested         ->positive
 [4] jimmie         ->johnson
 [5] top            ->story
 [6] secret         ->service
 [7] test           ->positive
 [8] mount          ->rushmore
+ ... omitted several edges
```

図８　ネットワークの表示
注．下記のスクリプトにもとづく．

パイプを使ってこれまでの一連の作業をスクリプトに書くと，次のようになります．

```
tw_fig <-tw_rank %>%
filter(n>20) %>%                    #　単語ペアの出現頻度を20を超えるものに指定
graph_from_data_frame()             #　ネットワークに変換
```

図８においてみられるように，このスクリプトを実行すると，たとえば，whiteからhouseへとネットワークが作成されています．igraphの結果tw_figはplot()関数で簡単に図９に似たグラフを描くことができます．

```
plot(tw_fig)
```

より細かな指定もできますが，ここではggplot2のグラフィック文法にしたがったパッケージggraphを利用して，ネットワークを描画します．最初にこのパッケージをインストールし，利用できるようにしておきましょう．

```
install.packages(“ggraph”)
library(ggraph)
```

それぞれ実行[Run]しておいてください．ggraph()の基本的な用法は次のようになります．

```
ggraph(graph, layout = )
```

引数graphにはtbl_graphオブジェクトの名前を指定します．つまり，graph_from_data_frame()で作成したオブジェクトです．ここではtw_figです．引数layout=“ ”には，いくつか用意されているレイアウトを利用します．“fr”，“kk”などがあります．この例ではlayout = “auto”を使っています．

```
set.seed(2020)                                  # 乱数を指定
ggraph(tw_fig,layout = “auto”)+                 # レイアウトを描きます．
    geom_edge_link()+                           # リンクを描きます
    geom_node_point()+                          # ノード(節)を描きます
    geom_node_text(aes(label=name),repel = T)+  # ノードに単語を表示させます
theme_void()                                    #  テーマを選択します
```

Rでは擬似乱数がR起動時に初期化され，毎回異なった乱数が発生します．このため同じ関数を利用していても結果が変わる場合があります．そこで乱数を再現するために乱数の種を設定するset.seed()関数が用意されています．set.seed()で指定すれば，結果が変わるのを防げます．数字は任意の整数でかまいませんので，ここではset.seed(2020)としています．

グラフィック文法はggplot2に従っていますので容易に理解できると思います．なお，geom_node_text()関数の中の引数repel = TRUE(もしくはT) は文字の重なりを防ぐことを指定しています．

さらに見やすくするために，リンクを描くレイヤーにedge_alphaを追加し，バイグラムの出現頻度―n変数列―にあわせてリンクの透明度を変更する設定を行ないます．このレイヤーを次のように書き換えてください．

```
geom_edge_link(aes(edge_alpha=n),show.legend = F)
```

edge_alpha = nでリンクの色の濃さをその単語ペアの出現頻度nに応じて変化するように指定しています．つまり，単語のペアの出現頻度が多くなると，両者を結ぶリンクの線がより濃く表示されます．またnの水準を示す凡例が自動的に表示されますが，show.legend = Fによって表示されないようにしています．

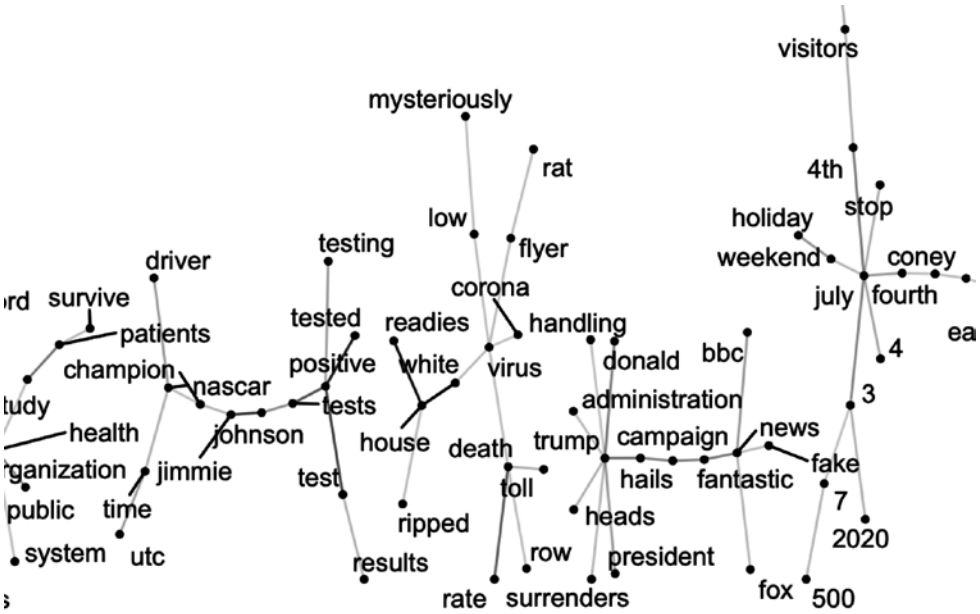

図９　新型コロナウィルスのツィートで見られる単語の関連
注．図の煩雑さを避けるためにlayout= “auto” によって強調表示された部分を掲載．

図９を見ると，いくつかの面白い結びつきがわかります．たとえば，図の左側から中央部分にかけてはpositiveとnascarネットワークに気づきます．これはNASCARのCupシリーズで７回のチャンピオンchanpionを獲得した経験のあるジミー・ジョンソンjimmie，johnsonが新型コロナウィルスに感染した，ということを示しています．また，中央部分のvirusを中心としたネットワークは，death，rateやwhite，low，mysteriolyと結びついています．このネットワークは白人の死亡率の低さが不思議だということを示唆しているのかもしれません．さらにトランプ大統領の選挙活動への注目もtrumpを中心としたネットワークから読み取ることができます．

Ⅳ－３　ツイッターテキストのセンチメント分析

これまでツイッターでつぶやかれた内容を分析することによって，新型コロナウィルスをめぐって「何」が注目されているのかを中心にみてきました．しかし，新型コロナウィルスをどのように感じ，どのように受け止めているのかには触れてきませんでした．そこで最後に，ツイッター

テキストにセンチメント分析を適用し，「社会」の気持ちをみることにします．

わたしたちは文書を読むとき，言葉がもつ感情的な意図を評価します．たとえば，それがポジティブなものなのか，ネガティブなものなのか，あるいは怒りや嫌悪なのかを考えます．センチメント分析とは，テキスト―ここではツイッターテキスト―に対する評価を，そのテキストを構成する単語をもとに評価する分析手法です．このためセンチメント分析には，言葉の持つ感情を評価した辞書が必要となります．その上で辞書と分析対象のテキストの単語を結合し，辞書にもとづいて対象のテキストを評価することになります．そこで最初に感情を評価するための辞書について触れておくことにします．

パッケージtidytextはsentimentsデータセットの中に次の3つの感情辞書を含んでいます．

- Bing[4]
  - 単語wordをポジティブかネガティブかに分類した辞書
- AFINN
  - －5から＋5の範囲で単語wordに正負のスコアを与えた辞書
- NRC
  - 単語wordをanger怒り，disgust嫌悪感，fear恐怖，joy喜び，trust信頼などに分類した辞書

最初に，簡単に，tidytextのBing辞書の中をみてみましょう．get_sentiments()関数は辞書データセットを取得するために利用されます．用法は次のとおりです．

```
get_sentiments(“利用する辞書名”)　# 辞書名には“bing”, “affin”, “nrc”と入力
```

スクリプトに次のように入力し，実行[Run]してみて下さい[5]．

```
get_sentiments(“bing”) %>%
        filter(sentiment == “positive”)
```

これを入力すると，filter()によってpositiveワードが抽出されます．

[4] Loughran-McDonald Sentiment lexiconも単語をポジティブ，ネガティブに分類した辞書ですが，これも利用可能です．

[5] はじめて利用する場合は，辞書をダウンロードするかどうかを尋ねられます．Yesの場合は1を入力し，ダウンロードして下さい．

```
# A tibble: 2,005 x 2
   word        sentiment
   <chr>       <chr>
 1 abound      positive
 2 abounds     positive
 3 abundance   positive
 4 abundant    positive
 5 accessable  positive
 6 accessible  positive
 7 acclaim     positive
 8 acclaimed   positive
 9 acclamation positive
10 accolade    positive
# … with 1,995 more rows
```

図10　辞書bingの中のpositiveワード

図10をみると，aboundから始まる単語のそれぞれが，positiveに評価されています．次に，bingデータセットに含まれるnegativeワードを見てみましょう．

```
get_sentiments(“bing”) %>%
        filter(sentiment == “negative”)
```

今度は図11のようにnegativeワードが表示されます．

```
# A tibble: 4,781 x 2
   word        sentiment
   <chr>       <chr>
 1 2-faces     negative
 2 abnormal    negative
 3 abolish     negative
 4 abominable  negative
 5 abominably  negative
 6 abominate   negative
 7 abomination negative
 8 abort       negative
 9 aborted     negative
10 aborts      negative
# … with 4,771 more rows
```

図11　辞書bingの中のnegativeワード

AFINN辞書は，辞書内の各単語に 正 または 負 のスコアを割り当てます．スコアがゼロより大きい場合はポジティブな感情を示し，ゼロより小さい場合は全体的にネガティブな感情を意味します．計算されたスコアがゼロの場合は，中立的な感情(肯定的でも否定的でもない)を意味します．

```
get_sentiments(“afinn”)
```

これによって図12のようなafinn辞書の中を見ることができます．

```
# A tibble: 2,477 x 2
   word       value
   <chr>      <dbl>
 1 abandon       -2
 2 abandoned     -2
 3 abandons      -2
 4 abducted      -2
 5 abduction     -2
 6 abductions    -2
 7 abhor         -3
 8 abhorred      -3
 9 abhorrent     -3
10 abhors        -3
# … with 2,467 more rows
```

図12　辞書afinnの中の感情評価

たとえば，abandon[放棄する]にはマイナス２の感情評価，abhor[忌み嫌う]にはマイナス３のスコアが与えられています．いずれもnegativeですが，negativeの程度はabondonよりもabhorが大きいということが理解されます．

こうした辞書にもとづいてわたしたちはツイッターテキストの中にpositiveワードが多いのか，negativeワードが多いのか，またその程度を評価できます．

Ⅳ－３－１　新型コロナウィルスのツィートにおいて何がもっともネガティブ(ポジティブ)な言葉か

それでは新型コロナウィルスを検索ワードに指定して収集したツイッターテキストに対して，Bingを使ったセンチメント分析を行なってみましょう．この例では取得したツイッターのテキストが単語に分割され，ストップワーズがすでに取り除かれ，text_twに保存されていることを想

定[6]しています．

最初にRパッケージdplyrによって提供されているinner_join()関数を使い，Bing辞書と結合します．inner_join()関数の書式は次のようになります，

```
inner_join(x,y)
```

この関数はx，y両方のデータフレームに存在する行だけを返します．この例ではtext_twとBing辞書に入っている単語wordが一致する行だけを返し，単語wordと感情評価sentimentの変数列からなるデータフレームを作成します．さらに，count()でwordとsentimentの数を数えます．一連の作業をパイプ，%>%でつなげて行います．

```
emotion <- text_tw %>%
        inner_join(get_sentiment("bing")) %>%
        count(word, sentiment,sort = TRUE)
```

これでツイッターのテキストに使われている単語に，positiveかnegativeかを割り当てたデータフレームemotionが作成されます．図13において結果が表示されていますが，たとえば，virusにはネガティブ評価，safeにはポジティブ評価が与えられています．

```
# A tibble: 2,354 x 3
   word     sentiment     n
   <chr>    <chr>     <int>
 1 virus    negative    742
 2 trump    positive    712
 3 death    negative    576
 4 crisis   negative    331
 5 toll     negative    295
 6 outbreak negative    280
 7 died     negative    239
 8 symptoms negative    236
 9 top      positive    227
10 safe     positive    224
# … with 2,344 more rows
```

図13　ツイッターテキストのセンチメント分析

[6] 付録のscript6にはツイッター情報の取得から記載されています．

図13のような表形式ではわかりづらいので，図13の結果を視覚的に表現したいと思います．ポジティブな感情とネガティブな感情を比較するために単語を並べてプロットしてみましょう．

このためにemotionを編集しなおします．手順は以下のとおりです．

1．positive，negativeといった感情別にグループ化します．

2．それぞれのトップ20の単語を取り出します．

以下のスクリプトはこれを実行するスクリプトです．

```
emotion <- emotion %>%
        group_by(sentiment) %>%          # 変数sentiment別にグループ化
        top_n(20) %>%                    # トップ20単語をピックアップ
        mutate(word = reorder(word,n))
```

次に，これをggplot2を使い，グラフにします．

```
ggplot(aes(reorder(word,n),n))+
        geom_col (show.legend = F)+
        facet_wrap(~sentiment, scales = “free_y”)+
        coord_flip()+
        theme_bw()
```

ggplot()関数の中のreorder()によってword変数を多い順に並び替えます．reorderの用法は次のとおりです．

```
reorder(並び替えの基準となる変数，並び替えの基準)
```

図14のように条件ごとに別々の図を描き並べる場合，facet_wrap()を使います．

```
facet(～条件，size= プロットのスケール)
```

“～”の後の条件はこの場合，sentiment変数を条件として使っています．この変数はpositiveかnegativeかの2種類の値を持ちますので，この2つによって区別されることになります．引数“scales =”はプロットのスケールを指定していますが，たとえば，“free”とした場合，プロットに

適したスケールが自動的に設定されます．“free_y”はy軸についてのみ自動設定を行うことを指定しています(x軸の場合は“free_x”です)．

coord_flip()関数は座標変換の命令です．つまり，縦軸と横軸を入れ替えてグラフ表示します．

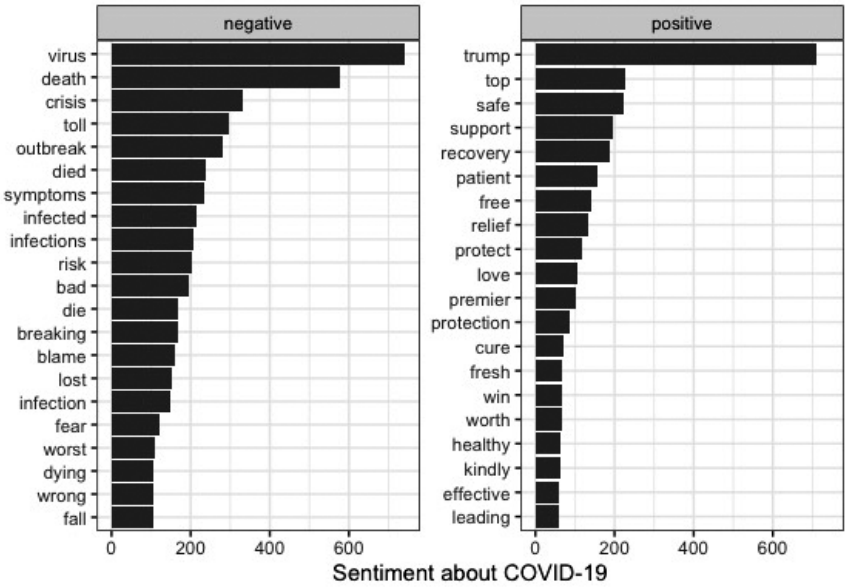

図14　COVID-19のツイッターテキストにおいて一般的なポジティブ・ネガティブ用語
注．ツイッターデータは７月７日16：00(日本時間)に取得

新型コロナウィルスをめぐって人々のネガティブな感情の形成に貢献しているのはvirus，death，crisis，outbreakという言葉・事実です．今後も，死亡者数と爆発的拡大が続けば，ネガティブな感情がさらに強まっていくことは確実です．

他方，ポジティブな感情の形成に貢献しているのはsafe，support，recoveryといった単語です．しかし，このケースでは，trumpがポジティブに記載されていることがわかります．ツイートの中で実際にこの言葉がどのように使われているのかを慎重に考察する必要があるかもしれません．ユニグラムを利用した分析ではストップワーズに入れておいた方が良いかもしれません．

Ⅳ－３－２　新型コロナウィルスのツィートにおいて何がもっとも一般的なangerもしくはdisgustな言葉か

新型コロナウィルスをめぐる人々のポジティブな感情，ネガティブな感情の形成にどのような言葉もしくは出来事が寄与しているのかを知ることができました．そこでさらに，ネガティブな感情に焦点を絞り，それがどのような言葉と結びついているのかを検討することにします．このためにNRC辞書を利用します．

```
get_sentiments("nrc")
```

このようにコンソール画面に入力すると，NRCの辞書の内容の冒頭部分が表示されます．

```
# A tibble: 13,901 x 2
   word        sentiment
   <chr>       <chr>
 1 abacus      trust
 2 abandon     fear
 3 abandon     negative
 4 abandon     sadness
 5 abandoned   anger
 6 abandoned   fear
 7 abandoned   negative
 8 abandoned   sadness
 9 abandonment anger
10 abandonment fear
# … with 13,891 more rows
```

図15　辞書NRCの中の感情評価

図15から理解されるように，NRC辞書はそれぞれの単語にtrust，fear等の感情表現を割り当てたものです．たとえば，abacus(算盤・そろばん)という単語にはtrust(信頼)という感情表現，abandon(放棄)にはfear(恐怖)やsadness(悲しみ)という感情が割り当てられています．

NRC辞書には，ネガティブな感情のラベルとしてanger，disgust，fear，sadnessの４つが利用されています．ここではこのうちfear(恐怖)とdisgust(嫌悪)に注目します．こうした２つのネガティブな感情がどのようなツイッターの単語と結びついているかをみてみましょう．

このために最初に，eomotion_nrcというデータフレームを作成します．手順は以下のとおりです．

１．ツイッターテキストの入ったデータフレームtext_twに，inner_join()を使ってnrc辞書を結合します．これによって２つの変数列word，sentimentを有するデータフレームが作成されます．

２．次に，２つの変数列を数えます．これで３つ目の変数nの列が作成されます．sort=Tで多い順に並び替えています．

３．filter()を使って変数列sentimentの値が"fear"と"disgust"である行を抽出します．

４．sentiment変数を使ってグループ別にまとめます．ここではfear，disgustの値の行しかあ

りませんので，fearとdisgust別にグループ化されます．

5．最後に，最初の20個の行だけを取り出します．

以上の結果がemotion_nrcに格納されます．以下が上の作業手順にしたがったスクリプトです．

```
emotion_nrc <-text_tw %>%
        inner_join(get_sentiments("nrc")) %>%
        count(word,sentiment,sort = T) %>%
        filter(sentiment %in% c( "fear","disgust")) %>%
        group_by(sentiment) %>%
        top_n(20)
```

View(emotion_nrc)でデータフレームの中を確認してみてください．ツイッターの中のどのような言葉がdisgust感情と結びつき，fear感情と結びついているのか理解できます．最後に，これを視覚的に表現しておきます（図16）．

```
ggplot(emotion_nrc,aes(reorder(word,n),n))+
        geom_col(show.legend = F)+
        facet_wrap(~sentiment,scales = "free_y")+
        coord_flip()+
        labs(y=NULL,x=NULL)+
        theme_bw()
```

新型コロナウィルスのツイッターテキストにおいて一般的なfear感情の言葉は，容易に想像がつきますが，パンデミックpandemicや死deathです．面白いのは政府governmentもfear感情に寄与しているようです．その他に多いのは医療関連の言葉―hospital，medical等―であることが分かります．他方，disgust感情の言葉は死deathや病気diseaseです．非難blameも無視できないようです．また作り話hoaxにもうんざりしていることが伺われます．

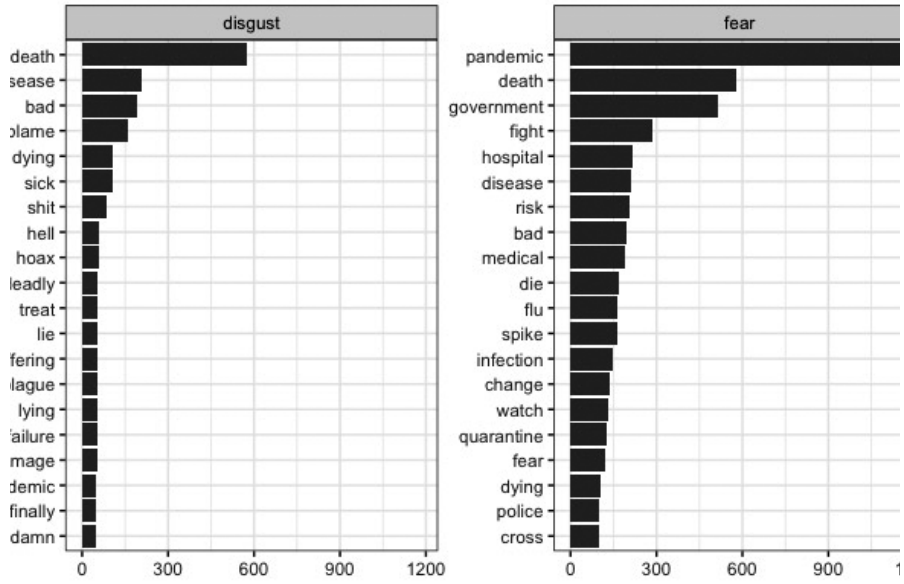

図16　COVID-19のツイッターテキストにおいて一般的なdisgust,fear感情用語
注．ツイッターデータは７月７日16：00(日本時間)に取得

Ⅴ．終わりに

これまで行なったテキストの分析手法は，もちろん，日本語テキストにも適用可能です．しかし，トークン化するにあたっても，センチメント分析を行うにあたっても，日本語辞書や感情表現辞書が必要となります[7]．すでにいくつかのそうした辞書が作成されています．またトークン化のツール，そのRに移植されたツールも開発されています．本資料ではまったく触れてきませんでしたが，基本的な分析方法は同じですので，そうしたツールを使って日本語テキストの解析にも挑戦してみてください．

【参考文献】

石田基弘『Rによるテキストマイニング入門』森北出版，2020年.

今井浩介『社会科学のためのデータ分析入門(下)』岩波書店，2018年.

Aksoy, C.G., Ganslmeier, M., and Poutvaara, P. (2020) Public Attention and Policy Responses to COVID-19 Pandemic, IZA DP No.13427.

[7] 日本語のテキストの分析にあたっては，RMecubを開発された石田基弘先生の『Rによるテキストマイニング入門』を参照して下さい．

Bali, R., Sarkar, D., and Sharma, T. (市川太佑・前田和寛・秋山孝史訳) 『Rではじめるソーシャルメディア分析―Twitterからニュースサイトまで』共立出版，2019年.

Silge, J. And Robinson, D. (大橋真也・長尾高弘訳)『Rによるテキストマイニング: tidytextを活用したデータ分析と可視化の基礎』オライリー・ジャパン，2018年. なお，bookdownで作成された本書の英語版(カラー版)はhttps://www.tidytextmining.comに公開されている.

【パッケージ一覧】

gtrensdsR, rtweet, tidytext, rm, wordcloud, ggraph, dplyr, tidyverse

【スクリプト一覧】

```
#=============================================================
#  script 1　ツイッター上のトレンドの取得と図1の作成
#=============================================================
library(tidyverse)
library(gtrendsR)

trend <- gtrends(keyword = c("COVID-19","PCR","AKB"),geo="JP",time="2020-01-21 2020-06-27")
summary(trend)
plot(trend)

#=============================================================
#  script 2　ツイッタートレンドの取得と図2の作成
#=============================================================
  jp_tr<- get_trends("japan")
  jp_tr %>%
    select(trend) %>%
    View()

#=============================================================
#  script 3　ツイッターデータの取得と図3の作成
#=============================================================
```

```
tweet_covid <- search_tweets(q="covid-19 OR coronavirus", n=18000,lang="en", include_rts = F)

tweet_covid %>%          #つぶやきの時間的推移のグラフ
  ts_plot("min")

#=========================================================
#  script 4　ツイッターテキストの編集とワードクラウド図5の作成
#=========================================================

# 変数created_atとtextを選択し，テキストを単語に分解
text_tw <- tweet_covid %>%
  select(created_at,text) %>%
  unnest_tokens(word,text)

# tidytextのデータセットstop wordsを読み込む
  data("stop_words")

#stop wordsを取り除く
  text_tw <-text_tw %>%
    anti_join(stop_words)

# filter()によってさらに不要なwordsを取り除く
no_words <- c("t.co","https","it's","1","2","20yearshuaweieurope","tech4all","ixqblwihjg","__ice9",
"sir","protocol","karachistockexchange","bbrightvc","sfpi5t5v7z","darknetflix","royalehighhalos",
"yellowstonetv","3ptmrwvz26","covid","19","amp","coronavirus","covid 19","people","covid19",
"covid_19","don't")

  text_tw <- text_tw %>%
  filter(!word %in% no_words)

# 単語wordの数を数え，多い順に並び替え，頻度が200を超えるもののみ抽出
text_tw<-text_tw %>%
```

```
  count(word, sort = TRUE) %>%
  filter(n >200)

# wordcloudを適用し，図５を描く
wordcloud(text_tw$word,text_tw$n,max.words = 100)

#====================================
#  script ５  ngram，図６，７の作成
#====================================

# created_at, text変数列を抽出，unnest_tokens()によってバイグラムを作成
text_tw04 <- tweet_covid04 %>%
  select(created_at,text) %>%
  unnest_tokens(biagram,text,token = “ngrams”,n= ２ )

# tidytextの中のストップワーズデータセットを読み込む
data(stop_words)

# その他の不要な単語を拾い出す
no_words <- c(“t.co”,“https”,“it's”,“1”,“2”,“20yearshuaweieurope”,“tech4all”,“ixqblwihjg”,“__ice9”,
“sir”,“protocol”,“karachistockexchange”,“bbrightvc”,“sfpi5t5v7z”,“darknetflix”,“royalehighhalos”,
“yellowstonetv”,“3ptmrwvz26”,“covid”,“19”,“amp”,“coronavirus”,“covid 19”,“people”,“covid19”,
“covid_19”,“don't”)

# バイグラムを２つの単語に分離し，いずれかがストップワーズであれば，filter()によって削除
tw <- text_tw %>%
  separate(biagram, c(“word1”,“word2”)) %>%
  filter(!word1 %in% no_words) %>%
  filter(!word2 %in% no_words) %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
```

```
# ワードの数を数え，多い順に並べる
tw_count<-tw %>%
  count(word1,word2,sort=T)

# 図9のためにネットワークを作成する
tw_fig <-tw_count %>%
  filter(n>25) %>%
  graph_from_data_frame()

# 図9を描く
set.seed(2020)
ggraph(tw_fig,layout = “auto”)+
  geom_edge_link(aes(edge_alpha=n),show.legend = F)+
  geom_node_point(size=1)+
  geom_node_text(aes(label=name),repel = T)+
  theme_void()

#================================================
#  script 6  Bing辞書を利用したセンチメント分析
#================================================

# rtweetパッケージのsearch_tweets()により，ツイッターデータを取得し，tw_covidに容れる
tw_covid <- search_tweets(q=“covid-19 OR coronavirus”, n=18000,lang=“en”, include_rts = F)

# 変数列textを選択し，テキストを単語に分割
tw_text <- tw_covid %>%
  select(text) %>%
  unnest_tokens(word,text)

# ストップワーズの除去
tw_text <-tw_text %>%
    anti_join(stop_words)
```

```
no_words <- c("t.co","https","it's","1","2","20yearshuaweieurope","tech4all","ixqblwihjg","__ice9",
"sir","protocol","karachistockexchange","bbrightvc","sfpi5t5v7z","darknetflix","royalehighhalos",
"yellowstonetv","3ptmrwvz26","covid","19","amp","coronavirus","covid 19","people","covid19",
"covid_19","don't","positive","negative")

tw_text <- tw_text %>%
  filter(!word %in% no_words)

# 辞書Bingを利用し，ツイッターのセンチメント分析
emotion <-tw_text %>%
  inner_join(get_sentiments("bing")) %>%
  count(word,sentiment,sort = T)

# 図14のプロットのために感情別にグループ化し，それぞれのグループの頻出頻度トップ20の単
語を抽出

emotion <- emotion %>%
  group_by(sentiment) %>%
  top_n(20) %>%
  mutate(word= reorder(word,n))

# 図14の作図
ggplot(emotion,aes(word,n))+
  geom_col(show.legend = F)+
  facet_wrap(~sentiment,scales = "free_y")+
  coord_flip()+
  labs(y="Sentiment about COVID-19",
       x=NULL)+
theme_bw()
```

```
#================================================
#  script 7  NRC辞書を利用したセンチメント分析
#================================================

辞書NRCを利用し，ツイッターのセンチメント分析
emotion_nrc <- tw_text  %>%
   inner_join(get_sentiments("nrc")) %>%
   count(word,sentiment,sort = T) %>%
   filter(sentiment %in% c( "fear","disgust")) %>%
   group_by(sentiment) %>%
   top_n(20)

# 図16の作成
ggplot(emotion_nrc,aes(reorder(word,n),n))+
geom_col(show.legend = F)+
   facet_wrap(~sentiment,scales = "free_y")+
   coord_flip()+
   labs(y=NULL, x=NULL)+
   theme_bw()
```