茨城大学教育実践研究 42(2023),28-43

データポータルを利用した簡便な出席データダッシュボードの作成（2）

—— shiny を用いたワードクラウドダッシュボードの作成 ——

林　延　哉*

(2023 年 10 月 23 日受理)

Creating a simple attendance data dashboard using a data portal (2): Creation of word cloud dashboard using shiny

Nobuya HAYASHI

キーワード：ワードクラウド，shiny，ダッシュボード

林(2022)では、毎回の授業で提出される出席データの情報を簡便に一覧するために、既存の BI ツールである Google データポータルを利用してダッシュボードを作成する方法を報告した。本報告では、先の報告で作成したダッシュボードと併用して用いる、出席フォームの自由記述欄に記入されたデータをワードクラウドとして表示するダッシュボードの作成について報告した。今回のワードクラウドダッシュボードの作成においては、Google データポータルではなく、統計処理環境 R とその IDE である RStudio、R の shiny パッケージを利用した。これらを用いることで、比較的簡単なプログラムによってインタラクティブなワードクラウド表示用ダッシュボードが作成できることを示した。

1 はじめに

筆者は林（2022）で、毎回の授業の際に提出される出席フォームに記入される情報を簡便に一覧するために、既存の BI ツールである Google データポータル（現 Looker Studio）を使ってダッシュボードを作成する方法を報告した。そのダッシュボードに表示したのは、出席者数や自由記述欄への書き込みの文字数のような数値化できるものと、自由記述内容そのものであった。

出席フォームへの当該授業内容についての自由記述欄に記載された内容については、当然のことながら一人ひとりの記入内容に目を通すわけだが、大人数の受講生がいる授業の場合、それ自体に

*茨城大学人文社会科学部（College of Humanities and Social Sciences, Ibaraki University, Mito, Japan）.

ある程度の時間が必要になる。

しかし、例えば、自由記述欄に記入された内容を一瞥することが可能であれば、受講生全体として授業内容がどの程度教員側の意図通りに伝わったか、あるいは異なる理解が行われたか等についての大まかな把握を、授業直後にも可能ではないかと考えられる。

そこで本報告では、先に報告したダッシュボードと併用することを想定した、自由記述欄の内容をワードクラウドを用いて視覚化するダッシュボードの作成について報告する。

2 今回利用するサンプルデータについて

本報告で使用する出席調査データ（以下サンプルデータと呼ぶ）は、林（2022）で使用したものと同じものである。筆者が過去の紙芝居に関する授業で実際に取得したデータから、一部の調査項目を削除し、学生番号、名前については架空のものに変換したものである。

Google フォームを利用して取得したデータは Google スプレッドシート上に蓄積される。サンプルデータは、そのことを想定して Google スプレッドシート上に入力されている。

サンプルデータの構成は表 1 のようになっている。この内、今回実際に使用するのは、B 列の「授業年月日」と E 列の「今日の授業について」になる。

「今日の授業について」は、当日の授業の要点などを、当日の授業の振り返りを企図して入力させるもので、字数制限のない自由記述としている。

サンプルデータは、講義回数 15 回、受講生は 76 名、全レコード数（データ行数）は 939 件である（より詳細な構成については林（2022）を参照されたい）。

表 1　サンプルデータの構成

列	ラベル	内容	フォーマット
A	タイムスタンプ	データが入力された時点での年月日時間。	YYYY/MM/SS HH:MM:SS
B	授業年月日	授業の年月日	YYYY/MM/SS
C	学生番号	学生番号	学生番号のフォーマット
D	名前	名前	名前
E	今日の授業について	当日の授業の要点、学んだこと、学ぶに当たって困難だったことなどを記入	自由記述（字数制限なし）

3 ワードクラウドについて

本報告では、出席調査データの自由記述欄に記入された内容を簡便に把握するための方法として、自由記述欄に記入された文章の単語の出現頻度のワードクラウドによる可視化を行う。

ワードクラウドでは、単語の出現頻度に基づいて、多数回出現する語を大きく、出現回数が少ない語を小さくした語の集まり（クラウド）として表示することができる（図 1）。

ワードクラウドはテキストマイニングのツールとして様々な場面で利用されているが、教育場面においても、例えば、授業中の掲示板書き込み内容の可視化（上野他，2019）、コロナ禍での実技科目の学生の学びの比較（菊池，2021）、音楽鑑賞教育における授業中の生徒の感想の共有（志村他，2022）等様々な場面で利用されている。

図 1 ワードクラウドの例

サンプルデータの「今日の授業について」は、当日の授業の要点などが記入されることを期待している。従って、当該授業の要点と考えられる語が多く出現していればある程度授業の意図が受け止められたことを想定できる。逆に、予想外の語の出現頻度が高いということであれば、教員側が想定していなかった訴求要素がそこにあったことが示唆されたと考えることもできる。最終的には一人ひとりの記述内容を精読していくことになるが、授業直後などに記述内容の様子を簡便に把握するためであれば、ワードクラウドはある程度有用であると考えられる。

4 今回使用する作成環境について

4.1 R 及び RStudio について

林 (2022) でダッシュボードの作成に使用した Google データポータル (現 Looker Studio) には、標準で用意されているワードクラウド用グラフは存在しない。また、ワードクラウドについては、単に単語の出現頻度に基づいて図を作成するだけでは、その内容を十分に把握できない場合もある。

例えば、サンプルデータは、全体として「紙芝居」をテーマとした授業のものであり、従って当然のことながら「紙芝居」という単語は頻出する。それをそのままワードクラウドにしてしまうと「紙芝居」が巨大な文字で表示され、他の語の殆どがその「影に隠れて」しまうような状態になってしまう（図 2）。その場合には「紙芝居」という語を除外することで、他の語がワードクラウド上に浮かび上がってくる（図 3）。個々の授業回にはそれぞれのトピックがあり、その都度除外したい語が出てくる可能性がある。また、表示される語の量を変化させることで特徴を把握することもで

図 2 「紙芝居」の出現頻度が他の語のそれを大きく上回っている

図 3 図 2 から「紙芝居」を除いてみる

きるので、作成された図を観ながら表示語数や削除語を変更し、それが即時に図に反映されることが望ましい。そうしたことを考慮して、ワードクラウドについては、R の shiny パッケージを RStudio 上で利用することとする。

R は、統計計算とグラフィックスのためのフリーソフトウェア環境[1]であり、そのルーツは 1984 年に開発された S 言語にあり（R 自体は 1993 年に登場している[2]）、数十年来用いられている定評のあるシステムである。

RStudio は、posit 社が R 向けに開発している IDE（Integrated development environment、統合開発環境）である。オープンソース版と商用版があり、オープンソース版は無償で利用できる[3]。RStudio を用いることで R の使い勝手は大きく改善される。

R や RStudio を利用するためには、Looker Studio とは異なり、利用するコンピュータにそれらをインストールする必要があるが、インストール自体は通常のアプリケーション同様それぞれのサイトから、インストーラをダウンロードして起動するだけのことなので、大きな困難はない。R については、CRAN（The Comprehensive R Archive Network, https://cran.r-project.org/）から、RStudio については、posit 社のサイト（https://posit.co/）からダウンロードできる。

なお、本報告では文脈によって「R」「RStudio」「R/RStudio」等と記載するが、RStudio 上で R を動作させていることを前提とする。

4.2 Shiny パッケージについて

上述の如く今回作成するワードクラウドでは、図を見つつ削除語や表示語数などを変更し、それが即座に図に反映するようなものを想定する。

このようなワードクラウド表示を実現するために今回は、R の shiny パッケージを利用する。

パッケージは、R に機能を追加するものである。R は長い年月に渡って利用されてきており、その間に蓄積されたパッケージが R の資産となっている。

shiny パッケージは、R でウェブアプリケーションを作成するためのフレームワーク[4]である。shiny パッケージを使うことで、表示された画面上で利用者の入力を、出力にリアルタイムに変更する仕組みを簡単に構築することができる。

もちろん、本報告では、ウェブアプリケーションとして公開することを目的とするのではなく、授業担当教員が授業のデータを観るために用いることを想定する。従って、作成したアプリケーションをウェブサーバー上で動作させる必要はなく、手元で使用している RStudio 上で実行する。

不特定多数の利用者を想定したウェブアプリケーションであれば、ユーザーインターフェースは重要な要素になり、また、ウェブサーバー上で常時稼働し常に最新のデータが反映されていることも重要な要素となるが、本報告では、あくまでも授業を担当する教員が、自身の授業についての受講生の反応を把握するために利用するものであるので、それらについては考慮しないこととする。

授業担当教員は、出席データの内容をチェックしたくなった際に、RStudio を起動し、これから作成するワードクラウドプログラムを実行することで、その時点で提出されている最新の出席データの内容をワードクラウドを用いて確認することができる、その際に、いちいちデータをダウンロードして読み込ませる必要はなく、Google スプレッドシート上の最新のデータにアクセスできる。

ユーザーインターフェースやリアルタイム性についてこだわらなければ、shiny パッケージを利用したインタラクティブなワードクラウド表示アプリケーションは、必ずしも R/RStudio という統計処理環境に精通する必要はなく、比較的簡単なプログラムの作成・実行によって実現できる。

以下、サンプルデータを使って、そのようなワードクラウド表示アプリケーションの作成の手順を示す。

5 ワードクラウド表示アプリケーションの作成

今回作成するアプリケーションの概要は以下の通りになる。

前述のように、アプリケーション全体の構築については先述の shiny パッケージを利用する。

ワードクラウドの作成については、wordcloud2 パッケージを利用する。

また、自由記述欄に記載された内容を単語に分割するために、R、RStudio とは独立したアプリケーションである MeCab と MeCab を R から利用するための RMeCab パッケージを利用する。

サンプルデータは Google スプレッドシート上に保存されているため、R からそれを呼び出すために googlesheets4 パッケージを使用する。

これらのパッケージを使いながら、次のような処理を行う。

1. 最新の出席フォームデータを Google スプレッドシートから R に取り込む。
2. 使用者が指定した授業日のデータを、1. で取り込んだデータから抽出する。
3. 2. で抽出したデータに対して形態素解析を行い入力内容を単語単位に分割した上で、それぞれの単語の出現頻度を計算する。
4. 3. で作成したデータに基づいてワードクラウドを作成する。その際、ユーザーインターフェースから入力される「表示する授業日」「表示する語の割合」「出現頻度の単位」「除外語」「フォントサイズ」「楕円率」を反映して、即時に図を変更できるようにする。

6 具体的なプログラム例

具体的なプログラムは RStudio を使って作成する。

RStudio で新規文書メニューから「Shiny Web App…」を選択すると、アプリケーション名を入力するダイアログが表示される。アプリケーション名は、これから作られるファイルが保存されるフォルダ名になる。なお、このダイアログでは「Application type」を選択するようになっているが、今回は「Multiple File」を選択することとする。

この指定が終わると、shiny パッケージを使ってウェブアプリケーションを実現するための server.R と ui.R の 2 つのファイルの雛形がフォルダ内に作成される。

今回は、これに加えて、global.R というファイルを同じフォルダ内に作成し、使用する。

作成したアプリケーションの実行については、上記のファイルのいずれかを RStudio で開いた際に表示される「Run App」ボタンを使って RStudio 上で実行する。

ui.R は、ユーザーインターフェースに関わるプログラムを記述するもので、今回の場合であれば、ワードクラウドの表示と、ワードクラウド描画に関する利用者による入力のための、ドロップダウンリスト・ラジオボタン・スライドバー・テキストフィールドの表示に関することを記述する。

server.R には、ワードクラウドとして表示するデータの抽出やワードクラウドの作成に関することを記述する。その際には、ユーザーインターフェースから入力された値を利用する。server.R で作成されたワードクラウドは、ui.R の指定によって表示される。

shiny パッケージで作成するアプリケーションは、server.R と ui.R の 2 つのファイルで動作させることができるが、今回は、server.R で使用するデータの読み込みや関数の定義を、server.R、ui.R が読み込まれる前に実行される global.R というファイルで定義する。こうすることで server.R 内の記述を簡素化でき、また修正も用意になる。以下、各ファイル内の記述について述べる。

6.1 ui.R

「Shiny Web App…」メニューを使って作成した ui.R ファイルには、既に、サンプルのプログラムが記入されている。「Run App」ボタンを使って実行すると、サンプルアプリケーションが実行できるようになっている。

新たに作成された ui.R にはコメントも含めて、30 行あまりのサンプルプログラムが記入されているが、動作に必要な部分だけを残すと、リスト 1 のような内容になる。

リスト 1　ui.R の骨格

```
library(shiny)
fluidPage(
      titlePanel("…")
      sidebarLayout(
            sidebarPanel(…),
            mainPanel(…)
      )
```

library(shiny)で、shiny パッケージを読み込んでいる。

fluidPage()は、ウインドウサイズに応じてレイアウトが変化するタイプのページデザインを設定する。

titlePanel()は、そのページのタイトルを記入する部分になる。

sidebarLayout()は、サイドバーを持つタイプのページレイアウトを設定し、sidebarPanel()でサイドバーの側を、mainPanel()で結果の表示部分の内容を設定する。

すなわち、これらでページ全体の雛形になっているので、後はその中に、実際に表示するものに関する記述を加えていけば、表示されるページのデザインを作成することができる。

今回は、この雛形をそのまま利用することにする。

左側のサイドバーには、ワードクラウドとして表示する「授業日」、表示する語の総語数に対す

図 4 作成する画面全体のイメージ

る「割合」、「除外する語」、ワードクラウドの「フォントサイズ」、ワードクラウドの「楕円率」、を指定するためのコントローラーを表示する。また、ワードクラウド上の文字の大きさは文字の頻度によって決まるが最も大きな度数と他の語の度数の差が大きすぎると殆どをその語が専有してしまうので、頻度を対数変換した数値を用意し、そちらを使って表示することもできるようにする。「授業日」にドロップダウンリスト、「表示する語の割合」「フォントサイズ」「楕円率」にはスライドバー、「出現頻度の単位」にはラジオボタン、「除外する語」にはテキストボックスを使用する。サイドバーの右側がワードクラウドを表示する領域になる（図 4）。

6.1.1 授業日

授業日については、サンプルデータから読みだした授業日を使って、ドロップダウンリストで選択できるようにする。ドロップダウンリストは、selectInput()という関数を使うことで作成できる。

```
selectInput(inputId = "date", label = "授業日", choices = 授業日_v,
selected = max(授業日_v))
```

inputId は、選択された項目が格納される変数を指定するものと考えればよく、date という名前で選択された内容が参照できる。label は画面に表示されるタイトル名、choices は実際の選択項目、selected は最初に選ばれている項目を指定している。choices に設定されている「授業日_v」は global.R で作成している変数で、中にはサンプルデータから読みだした授業日が格納されている。この処理は global.R で行っている。selected には、max(授業日_v)で、最新の授業日を設定している。これがデフォルトで表示される授業日になる。

6.1.2 表示する語の割合等

表示する語の割合は、スライドバーで設定する。スライドバーを作成するには、sliderInput()関

数を用いる。

```
sliderInput(inputId = "prop", label = "表示する語の割合", min = 0, max =
1, value = 0.3)
```

inputId と label は、selectInputx()と同様、スライドバーで指定された値が入力される変数と、画面に表示されるタイトルになる。min と max はスライドバーの最小値と最大値を指定している。value = 0.3 は初期値を 0.3 に設定している。

スライドバーについては、フォントサイズ、楕円率にも用いる。

```
sliderInput(inputId = "fontsize", label = "フォントサイズ", min = 0, max = 1,
value = 0.3)
sliderInput(inputId = "ellipticity", label = "楕円率", min = 0, max = 1,
value = 0.5)
```

6.1.3 出現頻度

出現頻度は、ラジオボタンで「raw（出現頻度）」「log（出現頻度を対数変換）」のいずれかを選択できるようにする。ラジオボタンは、radioButtons()関数で作成する。

```
radioButtons(inputId = "scale", label = "出現頻度の単位", choices =
c("log（出現頻度を対数変換）" = "log", "raw（出現頻度）" = "raw"), selected
= "log")
```

choices では選択肢を指定している。「"log（出現頻度を対数変換）" = "log"」という記述は、画面のラジオボタンには「log（出現頻度を対数変換）」と表示され、これを選択すると、「log」という語が選択されるように設定している。選択された「log」は、inputId で指定している scale の名前で参照できる。「"raw（出現頻度）" = "raw"」も同様に、こちらを選択すると「raw」が選択される。c()は、複数の項目をベクトルという R のデータ形式にまとめるための関数である。

6.1.4 除外する語

除外する語はテキストフィールドに入力することで指定できるようにする。複数項目を設定する場合は、カンマ、あるいはスペースで区切ることとする。

テキストフィールドは、textInput()を使って作成する。複数項目をカンマ、あるいはスペースで区切ることは画面にその旨を表示するのみで、特に入力チェックは行わないこととする。一般のアプリケーションに求められるような入力値のチェックやエラー回避のためのチェックを加えることよりも、できるだけ簡単なプログラムで、使用したい機能が得ることを優先する。カンマあるいはスペースで区切られた複数の語のそれぞれを独立して抽出する作業は global.R で行い、textInput()では、テキストの入力のみを行う。

```
textInput(includeId = "stopwords", label = "除外する語（複数の場合はカン
マ又はスペースで区切る）", value = "紙芝居")
```

「value = "紙芝居"」は、初期値として「紙芝居」を指定している。

6.1.5 sidebarPanel の構成

リスト2　ui.R の sidebarPanel()の設定内容

```
fluidPage(
   titlePanel("紙芝居論")
   sidebarLayout(
      sidebarPanel(
      selectInput(inputId = "date", label = "授業日", choices = 授業日
_v, selected = max(授業日_v)),
      sliderInput(inputId = "prop", label = "表示する語の割合", min = 0,
max = 1, value = 0.3),
      radioButtons(inputId = "scale", label ="出現頻度の単位", choices =
c("log（出現頻度を対数変換）" = "log", "raw（出現頻度）" = "raw"), selected
= "log"),
      textInput(inputId = "stopwords", label = "除外する語（複数の場合はカ
ンマ又はスペースで区切る）", value = "紙芝居"),
      sliderInput(inputId = "fontsize", label = "フォントサイズ", min =
0, max = 1, value = 0.3),
      sliderInput(inputId = "ellipticity", label = "楕円率", min = 0,
max = 1, value = 0.5)
   ),

   mainPanel(
      plotOutput("distPlot")
   )
 )
)
```

これらのコードを、sidebarPanel()内に記述する（リスト2)。titlePanel()には、ページのタイトルとしたい文字列を入力する。なお、この時点では、mainPanel()の中は、サンプルコードが残っているが、これは後に書き変える。

6.2 global.R

次に server.R で表示するワードクラウドを定義するのだが、その前に、server.R 内で使用するデータや関数を global.R 内で、事前に設定しておく。

global.R については雛形が作成されないため、RStudio の新規文書として R script ファイルを作成し、global.R のファイル名で ui.R、server.R と同じフォルダ内に保存する。

global.R 内では、以下の作業を行う。

・Google スプレッドシート上のサンプルデータを、R に読み込む。

・読み込んだサンプルデータから、授業日のリストを「授業日_v」と言う名前の変数にベクトル形式で保存する。これは、ui.R の授業日選択の部分の choices の設定に使用する。

また、以下の作業を行う関数を定義する。

・指定された授業日のデータを、読み込んだサンプルデータの中から抽出する。

・抽出したデータの自由記述欄の内容に対して形態素解析を行い、単語の出現頻度を計算する。
・計算された出現頻度表から、形態素解析結果の品詞細分類項目を用いて所定の単語だけを抽出する。また、同時に、除外語に指定された単語を除外する。頻度として用いる単位（実際の頻度か log 変換したものか）を設定して、ワードクラウド作成用データを作成する。

これらのことを記述するため global.R については、内容が多少複雑になるが、このことで、server.R 内の記述を簡素化することができる。

6.2.1 使用するライブラリの設定

global.R では、R に追加機能を提供するパッケージをいくつか利用している。まずそれらを利用するために library()関数を使って使用するパッケージを指定する。

```
library(shiny)
library(tidyverse)
library(googlesheets4)
library(RMeCab)
```

tidyverse パッケージはデータの扱いを容易にしたり、プログラムの可読性を高めることに寄与したりする様々な機能を提供するパッケージである。googlesheets4 パッケージは Google スプレッドシート上のデータを読み込むために使用する。RMeCab パッケージは、形態素解析ソフトである MeCab を R から利用可能とするライブラリで、そのために事前に MeCab をコンピュータにインストールしておく必要がある。インストールについては、RMeCab のオフィシャルサイトの記載に従えばよい（http://rmecab.jp/wiki/index.php?RMeCab）。これらをファイルの冒頭に記入しておく。

6.2.2 Google スプレッドシートの読み込み

Google スプレッドシート上のデータを R に読み込むには、googlesheets4 パッケージに含まれる read_sheet()関数を用いる。所定の Google スプレッドシート上のデータを、data という名前の変数に取り込む場合には、以下のような記述になる。

```
data <- read_sheet("読み込むスプレッドシートの URL")
```

引数のダブルクォーテーション内は、読み込むスプレッドシートの URL になる。

この命令は、実際にアプリケーションが実行される際には、Google スプレッドシートへのアクセスのためのログインを RStudio のコンソールで要求されるので、自分の使用しているアカウントでログインする。これで、アクセスが可能になる。

6.2.3 授業日ベクトルの作成

読み込んだデータは、表1に示した構造を維持したまま R のデータフレームとして読み込まれる。

このデータの授業年月日データから、それぞれの授業日を取り出し、ベクトルという R のデータ形式で変数に格納する。

プログラムでは、先に data に取り込まれたデータフレームから、授業年月日の列だけを取り出し、念のために欠損値（NA）を取り除いた上で、重複データを削除する。元々のデータフレームでは、1 行がある日のある受講生のデータなので、授業年月日には、その回の出席者の数だけ、同じ

授業年月日が入力されているためである。arrange(授業年月日)は、日付を昇順に並べている。

その上で、そのように整理した授業日データから、実際のデータをベクトル形式で取り出し、「授業日_v」という名前（名前は何でも良いが、ui.R の授業日の選択部分の choices に設定した名前と一致している必要がある）のオブジェクトに格納する。

この作業は、以下のように記述することができる。

```
授業日_df <- data |> select(授業年月日) |> na.omit() |> unique() |>
arrange(授業年月日)
授業日_v <- 授業日_df$授業年月日
```

6.2.4 抽出する品詞の設定

形態素解析を行うと、文章を単語に分割することができる。分割された単語の出現頻度を数えるわけだが、その際には単語として意味を汲み取りやすい品詞として名詞と形容詞のみを取り出すことにする。また、形態素解析に用いる MeCab は、文を単語に分割するだけではなく、品詞や品詞に関するより細かな分類（品詞細分類）を同時に出力する。そこで、ここでは品詞細分類が「一般」「自立」「固有名詞」に該当するものだけを抽出することにする。

この抽出の対象とする品詞の設定部分は以下のような記述になる。

```
include_pos1 <- c("名詞", "形容詞")
include_pos2 <- c("一般", "自立", "固有名詞")
```

include_pos1 と include_pos2 という 2 つの変数に、採用する品詞・品詞細分類を格納している。

6.2.5 該当する授業日のデータを抽出する

授業日の選択に合わせて、該当する授業日のデータを抽出する関数を定義する。

関数は、元となるデータと、選択された授業日を与えられると、その授業日に該当するデータを出力する。

授業日は文字列で与え、これを日付型に変換して、データ内の授業年月日と比較して、一致したものを取り出す。日付型への変換には as.POSIXct()を使用している。

この関数の定義は以下の様になる。関数名は ext_date_data()としている。

```
ext_date_data <- function(df, date) {
  date <- as.POSIXct(date, tz = "UTC")
  res_df <- df |> dplyr::filter(授業年月日 == date)
  return(res_df)
}
```

使用する際には、引数の df としてデータフレーム名、date として抽出したい日付を表す文字列を与える。結果として、授業日が date と一致するデータだけが返される。

6.2.6 形態素解析を実行し頻度表を作成する

リスト 3　pp_for_wc()関数の定義

```
pp_for_wc <- function(df, prop, scale, stopwords) {
    # ui で入力された除外語 stopwords をベクトル化
    stopwords <- str_split(stopwords, pattern = c("[,、　]+"),
simplify = TRUE)
    # POS2 を使って語を抽出し、stopwords に含まれる語を削除
    ext_df <- df |>
        dplyr::filter(POS2 %in% include_pos2, !TERM %in% stopwords)
    # 作図に使う頻度の値の選択
    result_df <- if (scale == "log") {
        ext_df |> select(TERM, Row1log) |> slice_max(Row1log, prop
= prop)
    } else {
        ext_df |> select(TERM, Row1) |> slice_max(Row1, prop =
prop)
    }
    return(result_df)
```

上で定義した、ext_date_data()の出力を使って形態素解析を行い、単語の頻度表を作成する関数を定義する。

形態素解析と頻度表の作成には、RMeCab パッケージの docDF()を用いる。この関数はデータフレームとそのデータフレームの中で形態素解析の対象となるデータの列番号、対象とする品詞を指定すると、形態素解析を実行し、その上で頻度表を作成してくれる。

この段階で、品詞については抽出を行ってしまうことになるので、先に値を設定した include_pos1 の値を使って、docDF()の処理を行う。

docDF()は、「TERM」「POS1」「POS2」「Row1」という 4 つの列を持つデータフレームを出力するが、頻度の値は Row1 に格納されている。この値を対数変換した値を計算し、それを Row1log という列名でデータフレームに追加する。関数名は、make_freq_table()、引数の df には ext_date_data()の戻り値を与える。Row1log 列が追加された docDF()に処理結果が戻り値になる。

```
make_freq_table <- function(df) {
  res_df <- docDF(df, col = 5, type = 1, pos = include_pos1) |>
    mutate(Row1log = log(Row1 + 1))
  return(res_df)
}
```

6.2.7 ワードクラウド用のデータを出力する

上で定義した make_freq_table()関数の出力から、include_pos2 で指定した品詞細分類に該当する語を抽出し、加えて、利用者が「除外する語」として指定した語を除外し、ワードクラウドに使用する頻度の単位を選択した上で、表示する語の割合で指定した分だけを頻度の大きい語から抽出

して、ワードクラウドに使用するデータを出力する関数を定義する。

この関数には、データフレーム、表示する割合、使用する頻度の単位、除外する語を引数として与えることになる。

関数の定義はリスト3のようになる。関数名はpp_for_wc()としている。

ユーザーインターフェースの部分で入力される除外する語は、複数の場合、カンマまたはスペースで区切られている。

```
str_split(stopwords, pattern = c("[,、　]+"), simplify = TRUE)
```

は、全角又は半角のカンマ、全角の読点、全角又は半角のスペースの1つ以上の連続を基準として、与えられた文字列を分割し、ベクトルに変換している。

if文の部分では、scale（使用する度数の単位）が"log"の場合と"raw"に応じて、Row1かRow1logかを出力している。その際にslice_max()で、propに指定した割合分を上位から抽出している。

6.2.8 3つの関数をまとめる

ここまで定義した3つの関数を使って、最終的にワードクラウドの作成に必要なデータが得られる。そこで、この3つの関数をひとつにまとめた関数を作成する。

```
make_wc_data <- function(df, date, prop, scale, stopwords) {
    res1_df <- ext_date_data(df, date)
    res2_df <- make_freq_table(res1_df)
    res3_df <- pp_for_wc(res2_df, prop, scale, stopwords)
    return(res3_df)
}
```

関数名はmake_wc_data()としている。データフレーム、授業日、表示割合、使用する度数の種類、除外語を引数として取る。関数内では、まず、ext_date_data()で授業日データを抽出し、その結果を使って make_freq_table()で、形態素解析し頻度表を作成、作成された頻度表から、最終的にワードクラウド作成に使用するデータをpp_for_wc()で抽出し、その結果を返していている。

ここまでが、global.Rの内容になる。

6.3 server.R

server.Rにもすでにサンプルコードが入力されているが、その内容を必要な部分だけ残すと以下のようになる。

```
library(shiny)
function(input, output, session) {}
```

このfunction(input, output, session) { }の「{ }」内に、ワードクラウドを作成するためのプログラムを記述することになる。

ワードクラウドの作成には、wordcloud2パッケージを利用する。そのため、library(wordcloud2)を加える。

```
library(shiny)
```

```
library(wordcloud2)
function(input, output, session) {}
```

ワードクラウドは、wordcloud2 パッケージに含まれる wordcloud2()を用いて作成する。wordcloud2()の引数には、ワードクラウドにする単語と頻度のデータフレーム、描画の際のフォントサイズ、ワードクラウドの楕円率を与える。

単語と頻度のデータフレームは、global.R 内で定義した make_wc_data()関数を使って作成する。この関数には、元となるデータ、授業日、表示する単語の割合、表示する頻度の種類、除外する語を与えることになる。

この内データについては､global.R でサンプルデータを data という名前の変数に格納している。

リスト５　server.R

```
library(shiny)
library(wordcloud2)
function(input, output, session) {
    output$wcPlot <- renderUI({
        wc_df <- make_wc_data(data, input$date, input$prop, input$scale, input$stopwords)
        wordcloud2(wc_df, size = input$fontsize, ellipticity = input$ellipticity)
    })
}
```

表示する授業日は、ui.R の中でドロップダウンリストを作成し、そこから選択するようにしているが、選択された結果は date という名前で参照するように定義した。この date の内容を、server.R ファイルの中では input$date で参照することができる。同様に、表示する単語の割合 prop、表示する頻度の種類 scale、除外する語 stopwords も、それぞれ input$prop、input$scale、input$stopwords で参照できる。そこで、ここでの make_wc_data()関数は、

```
wc_df <- make_wc_data(data, input$date, input$prop, input$scale, input$stopwords)
```

のように書くことになる。関数の返すデータは wc_df という変数に格納している。この wc_df を使って、

```
wordcloud2(wc_df, size = input$fontsize, ellipticity = input$ellipticity)
```

とすることで、ワードクラウドを作成できる。size はフォントサイズ、ellipticity は楕円率を、それぞれ ui.R で設定したスライダーバーの値から得るようにしている。

server.R で作成したワードクラウドを、ui.R 側で表示できるようにするために、上の 2 つの命令をまとめて、ui.R 側で参照できるようにする。そのために renderUI()関数を使う。

それらを含めた server.R 全体のプログラムはリスト５のようになる。

6.4 再度、ui.R

server.R で作成したワードクラウドは、最終的に output$wcPlot に格納された(wcPlot は参照用の名称で、任意の名称を用いることができる)。これは、ui.R 側で「wcPlot」として参照することができる。

図５ 「猫三味線」の回のワードクラウド

すでに、ui.R では、サイドバーについては定義をしてある。残る mainPanel()に server.R で作成したワードクラウドを表示することになるが、server.R 側で renderUI()を使って作成されたグラフは、renderUI()と対となる uiOutput()を使って、出力することができる。uiOutput()の引数で、表示するグラフの名称を指定すれば良い。今回の場合は「wcPlot」になる。

ui.R の mainPanel()内に uiOutput()で表示するワードクラウドを指定する。

この部分の記述は以下のようになる。

```
mainPanel(
  uiOutput("wcPlot")
)
```

6.5 プログラムの実行

ここまでで必要なプログラムはすべて記述できたことになる。

当初作成したフォルダの中に ui.R、server.R、global.R の３つのファイルが作成されている。

３つのファイルのいずれかを RStudio で表示し、「Run App」ボタンをクリックすれば、アプリケーションが起動する。起動すると、初期値を利用してワードクラウドが表示される。

授業日・表示する語の割合・出現頻度の単位・除外する語・フォントサイズ・楕円率は、それぞれの値を変更すればリアルタイムでワードクラウドが再描画される。

図１は、初回ガイダンス時のデータを用いたもので、記述内容もガイダンスの内容を反映したものが中心になるが「よろしくお願いします」という慣用句の頻度が高いのは経験上頷ける。「楽しみ」という語の頻度も多いので、少なくともガイダンスに関しては、ある程度受講生の興味を喚起できたのではないか、と推測できる（図１)。

別のある授業日には『猫三味線』という街頭紙芝居の映像を観ているが、「映像」「面白い」「絵」などの語に混ざって「ディズニーランド」「カラオケ」と言った語も目立つ（図 5)。時代物の紙芝居であるが、演者のアドリブで「ディズニーランド」や「カラオケ」といった語が登場する。そうしたところに受講生が訴求していることが推察される。

7 終わりに

先の報告で示したGoogleが提供するLooker Studioを利用する方法に比べると、Rによるプログラムの記述が必要になる分手間がかかることは否めないが、一度作成してしまえば、僅かな修正で他の授業にも利用できるようになる。

前回の報告同様、今回の報告も、多忙な中で実際に授業に当たっている教員が、ある程度でも自らの授業の改善・検討を効率的に行えることに寄与することを目的としたために、具体的な作成方法やプログラムについての記述を中心とした。ツールのひとつとして関心を持っていただければと思う。

注

1) The R Foundation 「The R Project for Statistical Computing」(https://www.r-project.org, 2023年6月18日閲覧).
2) Wikipedia 「R言語」(https://ja.wikipedia.org/wiki/R言語, 2023年6月18日閲覧) .
3) posit 「RSTUDIO IDE The most trusted IDE for open source data science」(https://posit.co/products/open-source/rstudio/,2023年6月18日閲覧)
4) CRAN 「shiny: Web Application Framework for R」(https://CRAN.R-project.org/package=shiny, 2023年06月18日閲覧)

引用文献

林延哉, 2022, 「データポータルを利用した簡便な出席データダッシュボードの作成」『茨城大学教育実践研究』41, 茨城大学全学教職センター, 196-211.

菊池理恵, 2022, 「保育者養成校におけるコロナ禍での実技科目の学生の学びの比較について―テキストマイニングを用いた自由記述による感想文の分析から―」『研究紀要』43, 名古屋柳城短期大学, 63-72.

志村泉・山口星香・小野貴史, 2022, 「音楽鑑賞教育における自由記述文による知覚・感受側面の分析ーテキストマイニングツールを活用した鑑賞授業の実践をもとにー」『信州大学教育学部附属次世代型学び研究開発センター紀要 教育実践研究』21, 信州大学教育学部附属次世代型学び研究開発センター, 31-40.

上野将・市川尚・富澤浩樹・阿部昭博, 2019, 「ワードクラウドを用いた授業中の掲示板書き込み内容を可視化するシステムの開発と評価」『情報処理学会第81回全国大会講演論文集』1, 情報処理学会, 557-558.