R によるテキストマイニング入門–初級

石田 基広*

2009/11/24

目次

* ishida@rmecab.jp

1　本日の講座の流れ

午前１ テキストマイニングの準備
午前２ 日本語解析の実際
午後３ 基礎
午後４ 応用１
午後５ 応用２

講座で利用のスクリプトファイル ism20091124.R を http://groups.google.co.jp/group/ishidamotohiro に公開している．

1.1　[午前-0-1] テキストマイニングとは

一言でいえば，電子化されたテキストから，機械的に有益な情報を取り出す技術の総称である．

テキスト，あるいは文章というデータは，定型化することが難しく，従来の統計分析あるいはデータ分析では十分活用されてこなかったデータである．これを積極的に分析の対象とするのが，テキストマイニングである．

1.2　[午前-0-2] テキストマイニングで何ができるのか

具体的な応用としては

１） WEB 上の情報の分析（第６章）２） アンケート自由記述分の分析（教科書第７，８章）３） 文書の自動分類（第９，１０章）

などが思いつくが，他にもいろいろ応用範囲があるだろう．

たとえば，最近徳島大学で FD の事例がある．これなども，これまでは単に眺められるか，あるいは冊子の後ろにまとめて掲載されている程度であった．

1.3　[午前-0-3]「R による」

「R による」テキストマイニングである

本日の講義では，午後の部で実際の解析事例を，この場で分析しながら，説明する．

その前に，午前の講義では，テキストマイニングを実行するための準備ついて説明したい．

[午前１]「テキストマイニングの準備」では，テキストマイニングを実行する上で必要となる形態素解析について述べる．

[午前２]「日本語解析の実際」では，形態素解析を R で実行する方法について述べる．

2　[午前-1] テキストマイニングの準備

2.1　[午前-1-1] テキストを構造化

- 解読（個人の営み–再現不可能）

- 解析 （機械に任せられる–再現可能）
  - 節，文， **語**，**文字** に分解
- 分解結果の集計（頻度表）
  - 頻度行列（ターム・文書行列 / N-gram 行列）

テキストマイニングを行う場合，テキストを解読ではなく，解析することになる．「解読」は機械ではなく，個々人が行う営みで，再現不可能な処理である．一方，「解析」は一部をコンピュータに任せることができる．

テキストの解析とは，文字の連なりを「節」,「文」,「語」,「文字」に分割することである．分割した結果から，要素ごとに頻度表を構成することで，テキストは頻度表（行列）に変換される．

2.2 [午前-1-2] 自然言語は曖昧である

ただしテキストを分割するというのは，容易な作業ではない．

自然言語解析を行う場合，どうしても，この手の曖昧性を解消することはできない．特に，分析者が，手作業で分割を行う場合，再現性がなく，そもそもデータの妥当性に疑念が生じてしまう．

そこで，少なくとも分析の再現性を保証するため，ルールを定め，そのルールを機械的に当てはめることが考えられる．機械的に当てはめるというのは，コンピュータに任せることが可能である．

テキストを単語や文字に切り分けることであれば，既存のコンピューターソフトを利用することで実行することができる．

2.3 [午前-1-3] 語と文字を対象とした解析を行う

今回は「節」,「文」については取り上げない．本講義では，テキストマイニングとして，テキストを「語」および「文字」に分割し，これを集計し，データ分析の対象とすることを説明する．

そこで本講義の午前最初の部では，テキストを単語あるいは文字で切り分ける方法について説明する．

頻度表を作成する場合，個別の要素，すなわち文字，あるいは語がそれぞれがテキスト中に何回出現したかをはかる場合と，ある要素とある要素の連なりが何回現れたかをはかる場合がある．これを Ngram という．

ここで川端康成の冒頭部分を例に，文字の頻度表と，語の頻度表を掲載する．文字単位の処理であれば，R の基本機能を使って頻度表を作成できる．

文字単位であれば R の基本機能を使って頻度表を作成できる．

```
> kawabata <- "国境の長いトンネルを抜けると雪国であった。夜の底が白くなった。信号所に汽車が止まった。"
> kawabata.2 <- strsplit(kawabata, "")
> kawabata.3 <- table(kawabata.2)
> # 文字の頻度表
> data.frame(kawabata.3)
   char freq
1     。    3
2    あ    1
3    い    1
```

```
4     が     2
5     く     1
6     け     1
7     た     3
8     っ     3
## 以下省略 ##
```

受講者の方々がどの程度 R に慣れておられるのか分からないのですが，基本的には，コマンドラインで操作を実行することになります．これが R を利用するためのハードルになっていることは否めません．

じつは，いずれ Excel から操作できるように拡張する予定もあります．

大なり記号の右にコマンドを書き込んで Enter を押すと実行されます．矢印は，右の命令の結果を，左の変数に代入することを表します．

2.4 [午前-1-4] 形態素解析 1

日本語テキストを単語を単位に分割するには，形態素解析器を利用する必要がある（教科書第 4 章）．

文章を単語に分割するためのソフトがいくつか公開されているが，本講座では Google の工藤卓氏が開発された MeCab を利用する．「彼は R を使っている」という文章を形態素解析してみよう。

- 文を語（形態素）に分割
  - 形態素解析器 MeCab: `http://mecab.sourceforge.net/` ：工藤 拓 氏
  - 例文：「彼は R を使った.」

表層型	品詞, 品詞細分類 1, 品詞細分類 2, 品詞細分類 3, 活用形, 活用型, 原形, 読み, 発音
彼	名詞, 代名詞, 一般,*,*,*, 彼, カレ, カレ
は	助詞, 係助詞,*,*,*,*, は, ハ, ワ
R	名詞, 一般,*,*,*,*,*
を	助詞, 格助詞, 一般,*,*,*, を, ヲ, ヲ
使っ	動詞, 自立,*,*, 五段・ワ行促音便, 連用タ接続, 使う, ツカッ, ツカッ
た	助動詞,*,*,*, 特殊・タ, 基本形, た, タ, タ

表 2–1 mecab の出力

文章を単語に分割するためのソフトがいくつか公開されているが，本講座では Google の工藤卓氏が開発された MeCab を利用する．「彼は R を使っている」という文章を形態素解析してみよう。

画面および表 2–1 に表示されている品詞情報出力は IPA 素性辞書にそっているが，MeCab は任意の辞書を利用することができる．詳細は `http://mecab.sourceforge.net/learn.html` を参照されたい．他の辞書としては UniDic (`http://www.tokuteicorpus.jp/dist/`) が有名である．解析結果から，この文章は六つの語から構成されている．特に注意いただきたいのが，「使った」が形態素二つとして分析されていることである．

出力は MeCab が利用している辞書によるのだが，推奨されている IPA 辞書を使った場合，左から表層型，すなわち文中に現れた語形に，右が品詞の情報，活用する語については原型，そして発音が続く．

表層型	形態素情報
	彼は R を使う。
彼	名詞, 代名詞, 一般,*,*,*, 彼, カレ, カレ
は	助詞, 係助詞,*,*,*,*, は, ハ, ワ
R	名詞, 一般,*,*,*,*,*
を	助詞, 格助詞, 一般,*,*,*, を, ヲ, ヲ
使う	動詞, 自立,*,*, 五段・ワ行促音便, 基本形, **使う**, ツカウ, ツカウ
。	記号, 句点,*,*,*,*,。,。,。
	彼女も R を使った。
彼女	名詞, 代名詞, 一般,*,*,*, 彼女, カノジョ, カノジョ
も	助詞, 係助詞,*,*,*,*, も, モ, モ
R	名詞, 一般,*,*,*,*,*
を	助詞, 格助詞, 一般,*,*,*, を, ヲ, ヲ
使っ	動詞, 自立,*,*, 五段・ワ行促音便, 連用タ接続, **使う**, ツカッ, ツカッ
た	助動詞,*,*,*, 特殊・タ, 基本形, た, タ, タ
。	記号, 句点,*,*,*,*,。,。,。

表 2–2　頻度の集計

%iffalse

MeCab は文を形態素に分割し，形態素ごとに行をとり，表層型と品詞情報をコンマで区切って出力するので，いったん csv 形式のファイルとして保存し，これを Excel に読み込ませて集計することもできる．

この二つの文では，動詞の「使う」が二度出ている．しかし表層型としては「使っ」と「使う」と形が違う．原型としては．もちろん同じである．従って，頻度を数える場合，表層型として数えるのか，あるいは原型として数えるのかで，頻度が変わってくる．

この点を理解されていない読者から時々質問を寄せられる．

通常は，原型として数えるべきだろう．つまり「使う」の頻度は 2 となる．これを Excel で集計する場合は，８ 列目でソートして並び替えて集計することになるだろう．しかし煩わしい操作である．

そこで日本語解析からデータの整形までを同一の環境上で行いたい．データ解析の環境としては R を利用し，R から日本語解析を行うことができれば便利である．

%fi

2.5　[午前-1-5] RMeCab パッケージ の導入

形態素解析器 MeCab と R との間をやりとりし，また MeCab の出力を加工する **RMeCab** パッケージ (`http://rmecab.jp/wiki/index.php?RMeCab`) .

- Windows 版
  - MeCab 本体を導入 (p.49)
  - RMeCab_0.85.zip をメニューから導入

  - MeCab とともに配布される libmecab.dll を RMeCab に追加する必要がある．

```
> path <- paste(.libPaths(),"RMeCab","libs", sep = "/")
> file.copy("C:/Program Files/MeCab/bin/libmecab.dll", path )
```

- Mac および Linux
  - 32 bit, 64 bit バイナリを用意

```
> install.packages("RMeCab_0.85.tar.gz", destdir=".",  repos = NULL)
```

%iffalse

RMeCab というのは私が昨年作成した R 用のパッケージであり，MeCab とバッググランドで通信して，指定された日本語テキストに形態素解析を行った結果を受け取り，これを R のデータ形式に変換して利用できるようにします．

従って利用するにはコンピュータに MeCab がインストールされていることが必要になります．Windows の場合，工藤氏のサイトからインストーラーをダウンロードすればよいことになります．

Mac や Linux でも MeCab を自動インストールすることは可能な場合があるようですが，できれば MeCab ソースファイルをダウンロードして，コンパイルしてもらった方が無難です．コンパイル方法などについては，私のサイトに説明があります．

Windows の場合，MeCab 本体をバックグランドで連携させるためには，MeCab が位置する場所をパスとして指定する必要があります．あるいは MeCab のインストール先にある libmecb.dll というライブラリを RMeCab のフォルダにコピーします．たとえば，R を起動して，ここにあるコマンドを実行すればインストールは完了します．これを自動的に行うバッチファイルも公開してますが，実行していることは，単なるファイルのコピーなので，そう難しいことではありません．

Mac では R のメニューから，また Linux では次のコマンドを実行すれば RMeCab が R 用のライブラリにインストールされます．インストール先の指定によっては保存先が云々という警告が表示されこともありますが，これは標準ではない場所にインストールが行われたことを意味するだけなので，無視してかまいません．

RMeCab にどのような機能があるのか，教科書に書いてあるのですが，実は出版以降も，RMeCab の改良を続けておりまして，そのため，一部の関数の引数を修正したり，あるいは教科書には記載されていない機能などを付け加えております．そこで，ざっとではありますが，RMeCab の機能について説明させていただきます．

またテキストマイニングの結果は，数千あるいは数万行に及ぶデータになるので，これらを効率的に検索や加工する技術が必要になる．文字列を検索，加工する技術を正規表現という．

2.6 [午前-1-6] RMeCab パッケージの基本操作

2.6.1 [午前-1-7] RMeCabC() 関数

教科書 52 ページ．関数そのものに変更点はないが，R のベクトルについておさらいするため，簡単に述べる．

R にはベクトルという概念がある．ベクトルは数値や文字などを要素とし，添え字という概念を使って要素を取り出すことができる．

これを利用して RMeCab では，要素を形態素，そしてその要素の名前に品詞情報を与えている．ただしデフォルトではベクトルを束ねたリストという形式で出力する．このリストが要素として含むベクトルは，どれ

も要素が一つだけ（形態素）なので，リストをベクトルに変換することができる．

```
> res <- RMeCabC("すももももももものうち")
> unlist(res)
    名詞       助詞       名詞       助詞       名詞       助詞       名詞
"すもも"       "も"    "もも"       "も"    "もも"       "の"    "うち"
```

ベクトルに名前がついているということは，名前で要素を検索できるということである．

```
> res2 <- unlist(res)
> res2[names(res2) == "名詞"]
    名詞       名詞       名詞       名詞
"すもも"    "もも"    "もも"    "うち"
```

このような検索では == を二つ並べた演算子を使う．

names(res2) == ”名詞” という部分だけ範囲指定して実行すると，次のような結果になる．

すなわち入力文には四つ名詞がある．

```
> length(res2[names(res2) == "名詞"])
[1] 4
```

複数の要素と照合する場合は %in% 演算子を使う．

R は関数を通してデータを操作する．関数には引数を与える．たとえば RMeCabC() 関数の第一引数は文字列だが，第二引数を指定できる．第二引数に数値 1 を指定すると形態素原型を返す．（教科書 p.54 上）

```
> res1 <- table(unlist(RMeCabC("彼は R を使う。", 1)))
> res1
  。     R    は    を 使う    彼
  1     1     1     1     1     1
> res2 <- table(unlist(RMeCabC("彼女も R を使った。",1)))
> res2
  。     R    た    も    を 使う 彼女
  1     1     1     1     1     1     1
> res3 <- merge(data.frame(res1), data.frame(res2), by = "Var1", all = TRUE)
```

ここでは RMeCabC の出力をさらに表形式に変換している．

二つの表を merge という関数で一つにまとめるとこうなる．

2.6.2 [午前-1-8] RMeCabText() 関数

教科書 55 ページ．関数の機能に変更なし．

名前の示すとおりテキストを対象に形態素解析を行う．その出力は MeCab の解析結果をそのままリスト形式で返すだけである．リストの各要素は，表層型，品詞情報や原型，発音などである．

```
> res <- RMeCabText("yukiguni.txt")
```

No.	term	テキスト 1	テキスト 2
1	。	1	1
2	R	1	1
3	は	1	NA
4	を	1	1
5	使う	1	1
6	彼	1	NA
7	た	NA	1
8	も	NA	1
9	彼女	NA	1

表 2–3　文書・ターム行列

```
file = yukiguni.txt
> res
[[1]]
 [1] "国境"        "名詞"        "一般"        "*"           "*"
 [6] "*"           "*"           "国境"        "コッキョウ" "コッキョー"

[[2]]
 [1] "の"      "助詞"    "格助詞" "一般"    "*"       "*"       "*"       "の"
 [9] "ノ"      "ノ"

[[3]]
 [1] "長い"              "形容詞"              "自立"                "*"
 [5] "*"                 "形容詞・アウオ段" "基本形"              "長い"
 [9] "ナガイ"            "ナガイ"

### 以下略

> res2 <- lapply(res, "[[", 8) 各リストの 8 番目の要素が原型
> data.frame(table(unlist(res2)))
```

RMeCabC(), RMeCabText() 関数は MeCab の出力をほぼそのまま返すだけであり，このままでは使いにくい．もっとも RMeCabText() 関数は MeCab を出力をそのままリスト化しているので，ユーザーによる応用的な加工が可能ではある．

ここで示す lapply 関数は，リストの要素それぞれについて，指定された番号の要素を取り出す働きをする．

そこで MeCab の出力をもう少し工夫して出力する関数をいくつか用意している．

3　日本語解析の実際

テキストマイニングを実行する際に，実際に利用することになるであろう関数をいくつか紹介する．

3.1　[午前-2-1] 形態素解析の結果を加工

- アンケート結果の分析
  - 名詞や形容詞（内容語）に焦点
- 文体の解析
  - 句読点や助詞（機能語）に焦点
- テキスト分類
  - ある閾値を超えるタームに焦点

%iffalse

形態素解析を使うことで，テキストを単語で分割し，これを R に取り込むことができる．ただしテキストを分析する場合，形態素解析の出力すべてを利用するのではなく，分析目的を達成するために効果的な要素を取り分けることや，あるいは，形は異なるが，意味は同じ要素を統合するなどの編集が必要になる．

ユーザーの希望する形態素解析をえるには，MeCab の辞書に登録しておけばよい．あるいは R 上の検索機能を使って置換する．

同義語に関しては最近日本語 WordNet が公開された．WordNet の機能は R に取り込むことが可能なので，同義語をまとめるという処理も，R 上である程度自動的に行うことが可能である．

いずれにせよ，最終的には分析者自身が確認を行い，必要があれば手作業で修正する必要がある．これは面倒で，泥臭い作業である．ただ，テキストマイニングを行う上で避けられない作業である．

市販のソフトには，独自の辞書をマウス操作で構築するなどインターフェイスに工夫を凝らしている．が，いずれにせよ，これを完全に自動化することはできないので，人手で行う面倒な作業であることには変わりない．ただし，こうした作業を何回か繰り返しておけば，その経験や辞書がその後の分析で大いに役に立つ．

こうした作業については後の応用編でも取り上げる．

%fi

3.2　[午前-2-2] データファイルの種類

テキストマイニングのデータソースのファイル形式は，大きく二つに分けられるだろう．

- データファイル（csv ファイル）
  - `RMeCabDF()` 関数（教科書 p.60）
  - `docMatrixDF()` 関数（教科書 p.69）
  - `docNgramDF()` 関数（教科書に記載なし）
  - `docDF()` 関数 （教科書に記載なし）
- テキストファイル（単独のファイル -s-, あるいはフォルダ全体 -p- ）
  - `MeCabFreq()` 関数 -s-（教科書 p.56）
  - `docMatrix()` 関数 -s- と `docMatrix2()` 関数 -s-,-p-（教科書 p.62-72 から改編）

- Ngram() 関数 -s-（教科書 p.72）
- NgramDF() 関数 -s- と NgramDF2() 関数 -s-, -p-（教科書 p.75）
- docNgram() 関数 -p- と docNgram2() 関数 -s-, -p- （教科書 p.77）
- collocate() 関数 -s- と collScores() 関数（教科書 p.79）
- docDF() 関数 -s-,-p- （教科書に記載なし）

3.3 データファイルからの解析

以下は csv 形式のファイル (c:/data/photo.csv) を，表計算ソフト (Excel) ではなく，メモ帳で開いてみたところである．

```
ID,Sex,Reply
1,F,写真とってくれよ
2,M,写真とってください
3,F,写真とってね
4,F,写真とってください
5,M,写真とってっす
```

これは被験者の属性と，自由記述を一つのファイルのおさめた例である．このようなファイルを R ではデータフレームという形式で扱う．

3.3.1 [午前-2-3] RMeCabDF() 関数

教科書 60 ページ．教科書と変更なし．

データファイルの自由記述文を含む列だけを指定して解析を行う．解析結果は MeCab の出力をリストとして返したものである．

3.3.2 [午前-2-4] docMatrixDF() 関数

教科書 p.69．なお辞書ファイルあるいは MeCab のリーソスファイルをそれぞれ dic=ないし mecabrc=引数で指定できるようにしました．

データフレームからターム・文書行列を作成する．

ターム・文章行列とは，行あるいは列に単語を，列または行に元テキストをとり，交差するセルに，頻度をとったデータである．

またターム・文書行列では頻度に「重み」を加えることがある．重みについては，後で docMatrix 関数を説明する際に補足する．

3.3.3 [午前-2-5] docMatrixDF() 関数共起語行列

docMatrix 関数には，教科書の執筆後，機能を一つ追加している．すなわち共起語行列である．

3.3.4 [午前-2-6] docNgramDF() 関数

教科書に記載はありません．

データフレームの指定列から文字あるいはタームの Ngram 頻度行列を作成する出力が他の関数と異なり，

行に文書（被験者回答）列にタームとなっている．

```
> targetText <- "H18koe.csv"
> dat <- read.csv(targetText, head = T)
> res <- docNgramDF(dat[,"opinion"])
> nrow(res);ncol(res)
> res[1:10, 1000:1005]
       [障] [集] [雑] [離] [難] [雨]
 Row1     0    0    0    0    0    0
 Row2     0    0    0    0    1    0
 Row3     0    0    0    0    0    0
 Row4     0    0    0    0    0    0
 Row5     0    0    0    0    0    0
 Row6     0    0    0    0    0    0
 Row7     0    0    0    0    0    1
 Row8     0    0    0    0    0    0
 Row9     0    0    0    0    0    0
 Row10    0    0    0    0    0    0
# ...
> res <- docNgramDF(dat[, "opinion"], N = 2)
> nrow(res);ncol(res)
> res[1:10, 1000:1005]
       [が-楽] [が-標] [が-殆] [が-残] [が-気] [が-永]
 Row1        0       0       0       0       0       0
 Row2        0       0       0       0       0       0
 Row3        0       0       0       0       0       0
 Row4        0       0       0       0       0       0
 Row5        0       0       0       0       0       0
 Row6        0       0       0       0       0       0
 Row7        0       0       0       0       0       0
 Row8        0       0       0       0       0       0
 Row9        0       0       0       0       0       0
 Row10       0       0       0       0       0       0
# ...
> res <- docNgramDF(dat[,"opinion"], type = 1)
> res[1:10, 1000:1005]
      [浴衣] [海] [海中] [海外] [海岸] [海底]
 Row1       0    0      0      0      1      0
 Row2       0    0      0      0      0      0
 Row3       0    0      0      0      0      0
```

```
 Row4       0    0      0      0      0      0
 Row5       0    0      0      0      0      0
 Row6       0    0      0      0      0      0
 Row7       0    1      0      0      0      0
 Row8       0    0      0      0      0      0
 Row9       0    0      0      0      0      0
 Row10      0    0      0      0      0      0
# ...
> res <- docNgramDF(dat[,"opinion"], type = 1, N = 2)
> res[1:10, 1000:1003]
      [バイキング-どれ] [バイキング-以外] [バイキング-朝] [バイク-多い]
 Row1                  0                 0               0             0
 Row2                  0                 0               0             0
 Row3                  0                 0               0             0
 Row4                  0                 0               0             0
 Row5                  0                 0               0             0
 Row6                  0                 0               0             0
 Row7                  0                 0               0             0
 Row8                  0                 0               0             0
 Row9                  0                 0               0             0
 Row10                 0                 0               0             0
# ...
> res <- docNgramDF(dat[,"opinion"], pos = "名詞", type = 1, N = 2)
> res[1:3, 1000:1002]
      [レンタカー-カーナビ] [レンタカー-中心] [レンタカー-人]
 Row1                     0                 0               0
 Row2                     0                 0               0
 Row3                     0                 0               0
# ...
> res <- docNgramDF(dat[,"opinion"], pos = "名詞", type = 1, N = 3, weight = "tf*idf*norm")
> res[1:3, 100:102]
      [いつ-開発-自然] [いつ-風景-生活] [いつか-ツケ-沖縄]
 Row1                0                 0                  0
 Row2                0                 0                  0
 Row3                0                 0                  0
# ...
> rowSums(res^2 , na.rm = T)# 行（文書・被験者回答）の合計は 1
#  データに NA が含まれる場合や，minFreq 引数に 2 以上を指定した場合は出力には NA が含まれるの
で注意
```

3.4 テキストファイルからの解析

3.4.1 [午前-2-7] RMeCabFreq() 関数

教科書 56 ページ．教科書と変更はないが，若干補足する．

RMeCabFreq() 関数はテキストに出現する語（形態素）の数（頻度）を計算する．

```
      Term  Info1    Info2 Freq
1      ある 助動詞        *    1
2        た 助動詞        *    3
3        だ 助動詞        *    1
4        と   助詞 接続助詞    1
5        が   助詞   格助詞    2
6        に   助詞   格助詞    1
7        の   助詞   格助詞    1  # "国境の長いトンネル"
8        を   助詞   格助詞    1
9        の   助詞   連体化    1 # "夜の底"
... 以下略
```

MeCabFreq() 関数は，形態素原型を基準にカウントするが，同一の語形であっても品詞情報（二つある）が一致しない場合は，別々にカウントする．

語形と品詞第 1 情報は一致するが，品詞第 2 情報異なるようなケースを同一の形態素としてカウントしたい場合は，以下のようにする．

```
> aggregate(res$Freq, res[1:2], sum)
      Term  Info1 x
1        。   記号 3
2      長い 形容詞 1
3      白い 形容詞 1
4        が   助詞 2
5        と   助詞 1
6        に   助詞 1
7        の   助詞 2
8        を   助詞 1
... 以下略
```

3.4.2 [午前-2-8] 出力からの抽出

対象テキストの分量（文字数あるいは単語数）にもよるが，一般に RMeCabFreq() 関数の出力は数百，数千，数万になるであろう．そこで，結果を画面にすべて表示させるわけにはいかない．確認したい部分だけ抽出する，表示する方法を知ることが重要になる．

言語テキストでは，頻度が 1，つまりテキスト中に 1 回しか出現しない単語が半数を占める．RMeCabFreq()

関数から，頻度が 2 以上の単語のみを抽出するには，以下のように操作する．

```
> res[res$Freq >= 2,]
   Term  Info1  Info2   Freq
2    た 助動詞      *     3
5    が    助詞 格助詞     2
23   。    記号    句点     3
### 以下略
```

res オブジェクトはデータフレームであるが，その Freq 列に頻度（数値）情報があるので，その要素が 2 以上である行だけを出力している．

MeCabFreq() 関数の出力から特定の品詞情報に該当するものだけを抽出する場合は検索用の演算子 %*% を使う．

```
> res[res$Info1 %in% c("名詞","形容詞"),] #名詞か形容詞の場合を抽出
       Term  Info1    Info2 Freq
13 トンネル    名詞      一般     1
14     信号    名詞      一般     1
15     国境    名詞      一般     1
16       底    名詞      一般     1
17     汽車    名詞      一般     1
18     雪国    名詞      一般     1
19       夜    名詞 副詞可能     1
20       所    名詞      接尾     1
21     白い 形容詞      自立     1
22     長い 形容詞      自立     1
```

特定の文字が含まれる Term を抽出する方法の一つは grep() 関数を使う．

```
> res[grep("国", res$Term) ,]
   Term Info1 Info2 Freq
15 国境   名詞   一般    1
18 雪国   名詞   一般    1

> res[grep("国|い", res$Term) ,]   # 「国」あるいは「い」が含まれるターム
   Term  Info1 Info2 Freq
15 国境    名詞   一般    1
18 雪国    名詞   一般    1
21 白い 形容詞   自立    1
22 長い 形容詞   自立    1
```

3.4.3 [午前-2-9] ターム・文書行列の作成： docMatrix() 関数と docMatrix2() 関数

教科書 p.61．引数の変更と追加．また教科書出版時に用意した data フォルダファイル内のテキストファイルには若干修正を加えている．

- p.67 の sym 引数は廃止し，kigo 引数に変える
- ターム共起頻度行列の追加（co = 1 引数）

docMatrix() 関数と docMatrix2() 関数はファイルからターム・文書行列を作成する．二つの違いは，前者は頻度に関する情報を表す行が二つ追加されていることと，最小頻度の指定にある（教科書 p.68）．

ターム・文書行列とは，行に出現した単語，列にテキスト名をとり，行と列の交差するセルには，ある単語があるテキストで出現した頻度が記録される．ターム・文書行列は単語・文書行列などとも呼ばれる．

```
> res <- docMatrix("doc", kigo = 1) # 教科書 p.67
file = doc/doc1.txt #テキスト内容は出版時と異なります
file = doc/doc2.txt
file = doc/doc3.txt
Term Document Matrix includes 2 information rows!
whose names are [[LESS-THAN-1]] and [[TOTAL-TOKENS]]
if you remove these rows, run
result[ row.names(result) !=  "[[LESS-THAN-1]]" , ]
result[ row.names(result) !=  "[[TOTAL-TOKENS]]" , ]
> res
                docs
terms            doc1.txt doc2.txt doc3.txt
  [[LESS-THAN-1]]        0        0        0
  [[TOTAL-TOKENS]]      12       14       16 # ここに記号頻度が含まれている
  学生                   2        2        1
  私                     2        0        0
  良い                   2        0        0
  数学                   0        2        2
  彼女                   0        2        2
```

注意点は kigo 引数に 1 をセットすると，句読点などの記号を含めた合計が計算される．ただし，タームとして記号が含まれることはない．

pos 引数に「記号」を含めると，タームとして句読点が含まれ，総計にも，これらの記号の頻度が加わる．

文書・ターム行列では，頻度に重みを与えることができる（教科書 p.70）．重みはとは，文書のサイズが異なる場合の頻度比較や，すべての文書に共通して現れるような単語の重要性を考量するための方法である．

3.4.4 [午前-2-10] 共起行列の作成（co = 1 引数）

教科書には記載のない機能．行名のタームと列名のタームが共起したテキスト数を要素とする対称行列．

```
> res <- docMatrix("doc", pos = c("名詞","形容詞","助詞"), co = 1)
file = doc/doc1.txt
file = doc/doc2.txt
file = doc/doc3.txt
now making co-occurrence matrix: this costs time!
> res
      terms
terms  は 学生 私 良い の 数学 彼女 で を
  は     3    3  1    1  2    2    2  1  1
  学生   3    3  1    1  2    2    2  1  1
  私     1    1  1    1  0    0    0  0  0
  良い   1    1  1    1  0    0    0  0  0
  の     2    2  0    0  2    2    2  1  1
  数学   2    2  0    0  2    2    2  1  1
  彼女   2    2  0    0  2    2    2  1  1
  で     1    1  0    0  1    1    1  1  1
  を     1    1  0    0  1    1    1  1  1
```

3.4.5 Ngram の作成

Ngram とは文字，あるいは単語，品詞が N 個連なった列とその頻度である．

3.4.6 [午前-2-11] Ngram() 関数

教科書 p.72．なお辞書ファイルあるいは MeCab のリーソスファイルをそれぞれ dic=ないし mecabrc=引数で指定できるようにしました．

type 引数で文字 (type =0)，単語 (type =1)，品詞 (type = 2)．また N 引数で数を指定する．

3.4.7 [午前-2-12] NgramDF() 関数と NgramDF2() 関数

教科書 p.75．なお辞書ファイルあるいは MeCab のリーソスファイルをそれぞれ dic=ないし mecabrc=引数で指定できるようにしました．

データフレームの指定行を対象に，Ngram() 関数と同じ処理を行うが，ただし出力は N 個の独立した列を用意する．

3.4.8 [午前-2-13] docNgram() 関数と docNgram2() 関数

教科書 p.77．なお辞書ファイルあるいは MeCab のリーソスファイルをそれぞれ dic=ないし mecabrc=引数で指定できるようにしました．

3.4.9 [午前-2-14] docDF() 関数

教科書に記載はありません．万能関数です．

第 1 引数で指定されたファイル (フォルダが指定された場合は，その中の全ファイル)，あるいは第 1 引数で

データフレームを，また第 2 引数で列（番号あるいは名前）を指定して，Ngram データフレーム，あるいはターム・文書データフレームを作成する．タームの場合，第 2 品詞情報までを出力する．

なお [[LESS-THAN-1]] と [[TOTAL-TOKENS]] の情報行は追加されない

指定可能な引数は

- target 引数：ファイル名ないしフォルダ名, あるいはデータフレームを指定
- column 引数：データフレームを指定した場合に列（番号あるいは名前）を指定する
- pos 引数： pos = c(“名詞”, “形容詞”, “記号”) のように指定する．デフォルトは全品詞
- minFreq 引数：頻度の閾値を指定するが，docMatrix() 関数の場合とは異なり， 全テキストを通じての総頻度を判定対象とする．例えば minFreq=2 と指定した場合，どれか一つの文書で頻度が二つ以上のタームは，これ以外の各文書に一度しか出現していなくとも，出力のターム・文書行列に含まれる．（ちなみに docMatrix() 関数では，文書ごとの最低頻度であった．したがって，doc1 という文書で二度以上出現しているタームが，他の文書で一度しか出現していない場合，このタームは出力のターム．文書行列に含まれるが，doc1 以外の文書の頻度は一律 0 にされていた.）
- weight 引数： 重みを付ける． ”tf*idf*norm”などと指定
- Genkei 引数：活用語を原型 (0) にするか，表層形 (1) にするかを指定
- co 引数：共起行列の作成．docMatrix2() 関数の事例を参照のこと

```
> (res <- docDF("doc", pos = c("名詞","形容詞","助詞"), N=1, type = 1))
file_name =  doc/doc1.txt opened
file_name =  doc/doc2.txt opened
file_name =  doc/doc3.txt opened
number of extracted terms = 9
now making a data frame. wait a while!
*
  TERM   POS1     POS2 doc1.txt doc2.txt doc3.txt
1   で   助詞 接続助詞        0        0        1
2   の   助詞   連体化        0        2        1
3   は   助詞   係助詞        2        2        2
4   を   助詞   格助詞        0        0        1
5 学生   名詞     一般        2        2        1
6 彼女   名詞   代名詞        0        2        2
7 数学   名詞     一般        0        2        2
8   私   名詞   代名詞        2        0        0
9 良い 形容詞     自立        2        0        0

> (res <- docDF(target, col = 3))
number of extracted terms = 15
now making a data frame. wait a while!

   Ngram Row1 Row2 Row3 Row4 Row5
```

```
1     。     2     0     2     0     0
2     い     0     1     0     1     0
3     く     1     1     0     1     0
4     さ     0     1     0     1     0
5     す     0     0     0     0     1
6     だ     0     1     0     1     0
7     っ     1     1     2     1     2
8     て     1     1     2     1     1
9     と     1     1     2     1     1
10    ね     0     0     1     0     0
11    よ     1     0     1     0     0
12    れ     1     0     0     0     0
13    を     1     0     0     0     0
14    写     2     1     1     1     1
15    真     2     1     1     1     1
```

3.4.10 [午前-2-15] 語の共起の指標：collocate() 関数と collScores() 関数

教科書 p.79 から大幅に変更しました．さらに辞書ファイルあるいは MeCab のリーソスファイルをそれぞれ dic=ないし mecabrc=引数で指定できるようにしました．

```
> res <- collocate("kumo.txt", node = "極楽", span =3)
file = kumo.txt
length = 413
> nrow(res)
[1] 33
>
> res[25:nrow(res),]
           Term Before After Span Total
25         極楽     10     0   10    10
26           様      2     0    2     7
27         蓮池      0     4    4     4
28         蜘蛛      0     2    2    14
29         行く      1     0    1     4
30         釈迦      2     0    2     7
31           間      0     1    1     3
32 [[MORPHEMS]]     18    13   31   413
33   [[TOKENS]]     40    30   70  1808
```

4　テキストマイニングの基礎

基礎編では，形態素解析の結果がどのように使えるのか，いくつか実例を示す．
教科書として指定した『R によるテキストマイニング入門』において分かりにくい箇所について説明する．

4.1　[午後-3-1] Zipf の法則

4.2　[午後-3-2] 語彙の豊かさ

4.3　[午後-3-3] 頻度スペクトル

4.4　[午後-3-4] 句読点の分布

4.5　[午後-3-5] 語の共起関係

collocate() 関数を使って，太宰治の『走れメロス』を分析してみよう．教科書 p.79

```
> merosu <- RMeCabFreq("merosu.txt")
> merosu[merosu$Term == "友",]
> merosu[merosu$Term == "友人" |  merosu$Term == "友達",]  #  or  検索
> merosu[grep("友", merosu$Term), ] # grep を使った検索．  "友"を含む語
> merosu <- collocate("merosu.txt", node = "友", span = 3)
> merosu2 <- collScores(merosu, node = "友", span = 3)
> merosu2[order(merosu2$MI),] # "友"と共起する語を MI 値で比較
```

5　午後４： テキストマイニングの応用１

5.1　[午後-4-1] テキスト分析：WEB 情報の分析

携帯電話に関わる WEB 上の口コミ分析．

```
> phone <- RMeCabFreq("phone.txt")  # ファイルの読み込み
> phone2 <- phone[ (phone$Info1 %in% c("名詞",  "形容詞"  ,"動詞" ,  "助動詞"))
 &   !(phone$Info2 %in% c( "非自立" ,  "数" )) ,] # p.85 下と同じ処理
> phone2[1:3,]   # 最初の 3 行を確認する
> phone3 <- phone2[phone2$Freq> 2,] # 頻度が 2 より大きい場合
> phone3[rev(order(phone3$Freq)),]
```

抽出した結果にはあらゆる品詞が入っているので，取捨選択する．ここでは携帯電話の評価に関する記述に興味があるので，内容語を抽出する．すなわち名詞や形容詞，そして動詞などである．

ただし名詞と分類されるものでも，「こと」のように，評価には直接関係ない形態素もある．そこで品詞第２情報を使って，これらの語をのぞく．すなわち非自立語と数詞をのぞく．

以上のように抽出した結果をもとのテキストと照合したい．R に取り込んでいるのは，もとのテキストを加工した結果である．別のソフト（Word など）を起動して元ファイルを表示し，検索をかけて確認する方法もある．が，ここでは R から元テキストをそのまま取り込んで照合してみたい．

```
> phoneRaw <- readLines("phone.txt") # テキストの単純な読み込み
> length(phoneRaw)
#   [1] 100                # phone.txt は 100 行からなる
> phoneMorp <- list(100)         # なので 100 行分の保存先を用意し
> for(i in 1:100){
     phoneMorp[[i]] <- unlist(RMeCabC(phoneRaw[i]))
     if(any( phoneMorp[[i]] %in%   c("売れる", "使える", "ない" ))){
          # print(as.vector( phoneMorp[[i]]))
          print(phoneRaw [i])
     }
 }
```

5.2 [午後-4-2] アンケートデータの分析

教科書 p.119 「沖縄観光アンケート」の分析

データには，観光客の性別，年齢，出身都道府県，自由記述回答が含まれる．ここでは年代と性別の組み合わせを要因としたとき，アンケートへの回答に傾向があるか調べてみる．

5.2.1 分析手順１：データの読み込み

まずデータそのものを読み込み，このうち自由記述回答文（第 1 列）について形態素解析にかける．

この結果，データそのもの (okinawa オブジェクト) に加えて，データに対応した形態素解析の結果 (res オブジェクト) の二つが生成される．二つは，行番号で対応させられる（無回答の被験者の場合，自由記述の解析結果の該当行には NA が登録されている）．

```
> okinawa <- read.csv("H18koe.csv")       # 教科書 p.119
> res <- RMeCabDF(okinawa, "opinion", 1) # 教科書 p.121
```

5.2.2 分析手順２：データの概観

教科書 p.121.

まず，形態素解析の結果を全体的にみておく．

```
> length(res) # 331 は被験者数．全員が回答しているわけではない
[1] 331
> length(unlist(res))# 総トークン数
[1] 14465
> length(unique(unlist(res)))# 総タイプ数
```

```
[1] 1996
> res.t <- table(unlist(res)) #総タイプ数の頻度表を作る
> length(res.t)## 1966
[1] 1996
> res.t[ rev(order(res.t)) ][1:10]  # 最頻出語
  。   の   が   た   、   て   に   を する   は
 823  727  631  510  485  447  435  339  322  314
```

最頻出語には助詞などの機能語が多い．これらは書き手の判別などの目的には有用な情報となるが，ここでの分析には必ずしも必要ではない．そこで，抽出すべき名詞を名詞，形容詞などの「内容語」に限定してみる．

5.2.3 分析手順３：内容語の抽出

教科書 p.122.

```
> res2 <- list()
> for(i in 1:length(res)){
     res2[[i]] <-  res[[i]][names(res[[i]]) == "名詞" |
                    names(res[[i]]) == "形容詞"]
 }
> res.t2 <- table(unlist(res2))
> res.t2[ rev(order(res.t2)) ][1:10]

  沖縄   観光     の     人     海   こと ほしい   良い   旅行   バス
   148    117     85     75     64     60     54     52     52     49
```

5.2.4 分析手順４：性別と年齢ごとの回答を抽出

教科書 p.123.

```
> res60F <- list()
> for(i in 1:length(res2)){
  if(!is.na(okinawa$Sex[i]) & okinawa$Sex[i] == "女性" & !is.na(okinawa$Age[i]) &
 okinawa$Age[i] == "６０代"){
    res60F[[i]] <-  res2[[i]]
  }else{  res60F[[i]] <- NA }
}
> res.t2 <- table(unlist(res2))
> res.t2[ rev(order(res.t2)) ][1:10]
#  沖縄   観光     の     人     海   こと ほしい   良い   旅行   バス
#   148    117     85     75     64     60     54     52     52     49
## 沖縄と観光，旅行 は除いた上位 10
> res60F1 <- unlist(res60F) [unlist(res60F) != "沖縄" & unlist(res60F) != "観光"&
```

```
 unlist(res60F) != "旅行"]
> res60F.t <- table(res60F1)
> res60F.t <- res60F.t [rev(order(res60F.t))][1:10]
> res60F.t
## ホテル   今回   バス     海     さ   良い   ない   時間     人   自然
##     17     12     12     11     11     10     10      9      8      8
```

これを性別と年齢の組み合わせごとに繰り返す

5.2.5 分析手順５：各世代・性別ごとの頻出語

教科書 p.124.

ここからさらに泥臭い作業になる．すべての世代・性別層に現れた最頻出語をまとめると以下のようになる．

```
> okinawa.lab <- unique(c(names(res20F.t), names(res20M.t), names(res30F.t), names(res30M.t),
 names(res40F.t), names(res40M.t),  names(res50F.t), names(res50M.t), names(res60F.t),
 names(res60M.t) ))
> okinawa.lab
 [1] "の"         "人"         "海"         "タクシー"   "多い"
 [6] "運転"       "ほしい"     "こと"       "いい"       "離島"
[11] "バス"       "充実"       "さ"         "良い"       "道"
[16] "車"         "者"         "きれい"     "ホテル"     "よう"
[21] "渋滞"       "交通"       "やすい"     "北部"       "那覇"
[26] "店"         "今回"       "日"         "等"         "前"
[31] "整備"       "自然"       "欲しい"     "モノレール" "私"
[36] "料金"       "美しい"     "方"         "道路"       "料理"
[41] "大変"       "ない"       "時間"       "必要"       "地"
```

5.2.6 分析手順６：頻出語の間引き

教科書 p.125.

このリストから，さらに要らないと分析者が判断するタームを除いてみる．

```
> oki <-  which(okinawa.lab %in%  c("さ", "の","こと","ない","私","者", "よう", "等",
"前","日", "那覇",  "今回", "方","大変","地", "ー"） )
> okinawa.lab <- okinawa.lab [ -oki ]
> okinawa.lab
##  [1] "人"         "海"         "タクシー"   "多い"       "運転"
##  [6] "ほしい"     "いい"       "離島"       "バス"       "充実"
## [11] "良い"       "道"         "車"         "きれい"     "ホテル"
## [16] "渋滞"       "交通"       "やすい"     "北部"       "店"
## [21] "整備"       "自然"       "欲しい"     "モノレール" "料金"
```

```
## [26] "美しい"     "道路"       "料理"       "時間"       "必要"
```

5.2.7 分析手順７：対象語の統合

教科書 p.125.

```
> res60F1[which(res60F1 == "きれい")] <- "美しい"
> res60F1[which(res60F1 == "欲しい")] <- "ほしい"
> res60F1[which(res60F1 == "いい")] <- "良い"
> res60F.t2 <- table(res60F1)
> res60F.t3 <- res60F.t2[names(res60F.t2) %in% okinawa.lab]
> res60F.t3
##   ほしい   やすい タクシー     バス   ホテル     運転       海     交通
##        6        2        1       12       17        5       11        1
##     時間     自然       車     渋滞       人     多い       店     道路
##        9        8        3        1        8        2        4        1
##   美しい     必要     料金     料理     良い
##        7        1        1        5       14
```

これをすべての性別・世代グループに行う.

5.2.8 分析手順８：タームの整理結果から改めてデータフレームを新規作成

教科書 p.126.

ここまでグループごとにばらばらにデータを整理したので，最後に再び一つにまとめる.

```
> okinawa.DF <- NULL
> okinawa.DF <- data.frame(word = names(res20M.t3), id = rep("20M",length(res20M.t3) ),
Freq = res20M.t3)
> okinawa.DF <- rbind(okinawa.DF,
      data.frame(word = names(res20F.t3), id = rep("20F", length(res20F.t3)),
 Freq = res20F.t3),
      data.frame(word = names(res30M.t3), id = rep("30M", length(res30M.t3)),
 Freq = res30M.t3),
      data.frame(word = names(res30F.t3), id = rep("30F", length(res30F.t3)),
 Freq = res30F.t3),
      data.frame(word = names(res40M.t3), id = rep("40M", length(res40M.t3)),
 Freq = res40M.t3),
      data.frame(word = names(res40F.t3), id = rep("40F", length(res40F.t3)),
 Freq = res40F.t3),
      data.frame(word = names(res50M.t3), id = rep("50M", length(res50M.t3)),
 Freq = res50M.t3),
      data.frame(word = names(res50F.t3), id = rep("50F", length(res50F.t3)),
```

```
 Freq = res50F.t3),
      data.frame(word = names(res60M.t3), id = rep("60M", length(res60M.t3)),
 Freq = res60M.t3),
      data.frame(word = names(res60F.t3), id = rep("60F", length(res60F.t3)),
Freq = res60F.t3))
> okinawa.t <- xtabs(Freq ~ word + id, data = okinawa.DF)
> row.names( okinawa.t )
##  [1] "ほしい"     "やすい"     "バス"       "ホテル"     "モノレール"
##  [6] "運転"       "自然"       "車"         "充実"       "人"
## [11] "整備"       "多い"       "道"         "道路"       "必要"
## [16] "料理"       "良い"       "タクシー"   "海"         "交通"
## [21] "時間"       "渋滞"       "店"         "美しい"     "北部"
## [26] "離島"       "料金"
```

5.2.9 分析手順９：対応分析の実行

教科書 p.127.

対応分析とは，カテゴリの水準ごとの相関の強さを求め，これをプロットとして付置することで，カテゴリ水準の関連性を数値あるいはグラフとして明らかにしようとする手法である．

```
>  library(MASS)
> okinawa.corr  <- corresp(okinawa.t,  nf = 2)
> biplot( okinawa.corr )
```

5.3 [午後-4-3] テキスト分類：読売新聞 WEB 版記事の自動分類

大量のテキストを対象に，これを何らかの基準で分類するという課題がある．テキストマイニングでは，テキストのテーマや内容を類似度に基づいて，テキストをいくつかの集合に分けるという課題が考えられる．

データマイニングの領域では，観測データをグループ分けする手法として，クラスター分析がよく用いられている．これはデータ間の距離を求め，この距離に基づいて観測値をまとめていくという手法である．

テキストマイニングにおいても，テキストを文字や形態素に分解し，その頻度にまとめたデータはクラスター分析の入力として用いることができる．

```
> DM <- docMatrix("c:/data/yomi",   pos = c("名詞","動詞","形容詞"))
> DM <- DM[ row.names(DM) !=  "[[LESS-THAN-1]]" , ]
> DM <- DM[ row.names(DM) !=  "[[TOTAL-TOKENS]]" , ]
> DM[100:105,]
> nrow(DM)
> ncol(DM)
> plot( hclust(dist( t(DM)))) # テキストの図
> plot( hclust(dist( t(DM), "canberra"), "ward")) # 方法を変える
```

この場合，単純に頻度そのものを利用したが，テキストマイニングでは潜在的意味インデキシングという手法を用いることもできる．

潜在的意味インデキシングでは特異値分解という技術が利用される．特異値分解とはターム・文書行列を，タームの自己相関行列，また文書の自己相関行列，そして次元の重要度を表す特異値の三つに分ける．

u 左特異行列 単語 x 単語 word vectors 語の共起頻度を表現している．もとのベクトルの要素が 0 であった場合にも関連性を表す数値が求められている

v 右特異行列は文書 x 文書行列 document vectors テキスト間における共起語の出現頻度を表現している．もとのベクトルの要素が 0 であった場合にも関連性を表す数値が求められている．

d 特異値は次元の重要度を表現している．

潜在的意味インデキシングの結果を使って，もとの文書を 3 次元空間にマッピングしてみる．3 次元グラフには，rgl パッケージを使う．

5.4 [午後-4-4] 作家判別： 森鴎外と夏目漱石の分類

文学的な主題として，書き手の特徴，あるいは癖をデータから見つけ出すという課題を考える．

ここでは書き手の特徴が表れやすいとされる文字のバイグラムの頻度をクラスター分析で分析することを考える．

ただ人文系の研究者からみると，これだけでは解釈に困ると思う．そこで，ここでは助詞と読点の組み合わせに書き手の癖が出るという，金・村上の研究結果を利用する．

これらの助詞と読点の頻度情報を圧縮した主成分を使うことによって，2 次元で二人の作家を判別することが可能になった．

6 午後５：テキストマイニングの応用２

ここからは同志社大学 教授 金 明哲 著 『テキストデータの統計科学入門』岩波書店 ほかに題材を求め，R および **RMeCab** パッケージ を利用したテキストマイニング技法について説明する．

6.1 [午後-5-1] ネットワーク分析１

テキストを分析するには，語や文字の頻度だけではなく，文字と文字，あるいは語と語のつながりや共起関係を分析することも重要である．Ngram はその方法の一つであったが，語と語のつながりをネットワーク図としてグラフ化して，分析者の考察を助ける，それがテキストマイニングにおけるネットワーク分析である．

ネットワーク分析では 語は 点，あるいはノードとして扱われ，語と語のつながりを先，あるいは辺として結ぶ．

麻生元総理と，鳩山総理の所信表明演説から，頻度の高い形態素を抽出し，そのネットワークを描いてみる．ネットワーク図を参照することによって，主観的ではあるが，二人の総理の政治姿勢が見えてくるであろう．

6.2 [午後-5-2] ネットワーク分析２

これはそれぞれののテキストを別々にネットワーク図にしたものであるが，複数のテキストと，そこに表れるタームの関連性を一枚のネットワークズに描くことも可能である．

6.3 [午後-5-2] 係り受けについて

6.4 [午後-5-4] 欧米語テキストのマイニング

最後に，簡単に欧米語のテキストを R でマイニングする方法について説明する．