KH Coder 3 Reference Manual

Koichi HIGUCHI[*1]

August 3, 2017

[*1] Ritsumeikan University <ko-ichi@ss.ritsumei.ac.jp>

Contents

Part A

KH Coder Reference Manual

Introduction

The basic procedures for quantitative text analysis using KH Coder can be learned using the Quick Start Tutorial. However, to carry out analyses with your own data, more detailed descriptions of its functions may be required. This Reference Manual provides users with such information.

Before preparing new text data to be analyzed, We reccomend you read through section A.2.1 to ensure your analyses progress smoothly. Similarly, before creating coding rules, consult the descriptions in section A.2.5. We also suggest reading through Section A.2 before starting your analysis.

From Section A.3 onward, there are definitions of each command, in which you will occasionally see paragraphs named "Internal processing". If you use the commands provided by KH Coder as they are, you may skip these paragraphs. As outlined in section A.13.1, KH Coder is an open source software, which can be modified and upgraded by individual users. Section A.11.1 describes how data stored in the databases (MySQL) by KH Coder can be retrieved directly, allowing more flexible data searches. We outline the details necessary for carrying out such operations under the subsection on "Internal processing".

KH Coder supports Windows, Linux and Macintosh. Once installed, the specifications and operating methods are the same, regardless of the OS used. However, because the installation methods and configuration procedures are somewhat different for each platform, this manual provides separate information for Windows and Linux in Section A.1: Installation and Start-up. The installation method for Macintosh is almost the same as Linux, but its configuration procedure is more complex. We now distribute automatic set-up software for Macintosh, as part of our paid support available through the official website.

A.1 Setup

A.1.1 Requirements

For Windows

To use KH Coder on Windows, a PC with Windows Vista or a later version is required. All software, such as ChaSen, MySQL, Perl and R, necessary to work with KH Coder is provided in the installation package for Windows. The package carries out all relevant configurations automatically. To analyze English text data, Java also must be installed.

For Linux

To use KH Coder on Linux, MySQL and Perl must be already installed. If the R software environment for statistical computing and graphics is not installed, then some of KH Coder's functions will not be available. To analyze Japanese text data, Chasen or MeCab is required. To analyze English text data, Stanford POS Tagger and Java need to be installed. The notes below are useful for setting up these

essential software on Linux.

■**ChaSen** The dictionary used in ChaSen (IPADIC) must be in the EUC-JP character encoding system. It is desirable to copy the whole dictionary folder of ChaSen into the home folder so that KH Coder can modify the settings of the dictionary, as required. After copying the dictionary, modify the settings file named "chasenrc" in the destination folder so that ChaSen refers to the copied dictionary. The "文法ファイル" (grammar file) statement in the "chasenrc" file specifies the directory of the dictionary. Normally, when ChaSen is installed on Linux, IPADIC is stored in the "dic" folder under /usr/local/share/chasen.

■**MySQL** First, you need to configure MySQL so it can correctly handle the EUC Japanese character sets. KH Coder 2.x converts Japanese text strings into EUC and then submits them to MySQL to search and retrieve data. Second, you need to configure access privileges in MySQL to allow KH Coder to carry out all operations, including database creation. To configure KH Coder, refer to "Manually setting up a connection to MySQL" under section A.1.3.

■**Perl** To use KH Coder on Linux, it is necessary to install various Perl modules, such as DBI, DBD::mysql, Jcode, and Tk. If these required Perl modules are missing, then the error message "Can't locate XX.pm in @INC (@INC contains: ...)" will be displayed when KH Coder is executed. The "XX" part indicates the name of the missing Perl module. Install the module as required. The current version of KH Coder was developed based on Perl 5.14.

■**R** Version 2.4 or a later version of R must be installed together with other necessary packages such as "maptools" and "scatterplot3d." As for Perl, if there are any missing packages, then an error message will be displayed. In such cases, install packages based on the resulting error messages. Many functions are unavailable without the installation of R, including:

- [Words] > [Descriptive Stats] > [Term Frequency Distribution] command plot function
- [Words] > [Descriptive Stats] > [Document Frequency Distribution] command plot function
- [Words] > [Descriptive Stats] > [TF-DF Plot] command
- [Words] > [Correspondence Analysis] command
- [Words] > [Multi-Dimensional Scaling] command
- [Words] > [Hierarchical Cluster Analysis] command
- [Words] > [Co-Occurrence Network] command
- [Words] > [Self-Organizing Map] command
- [Documents] > [Cluster Analysis] command
- [Coding] > [Crosstab] command chi-square test and plot function
- [Coding] > [Correspondence Analysis] command
- [Coding] > [Multi-Dimensional Scaling] command
- [Coding] > [Hierarchical Cluster Analysis] command
- [Coding] > [Co-Occurrence Network] command
- [Coding] > [Self-Organizing Map] command

A.1.2 Installation and start-up

On Windows

To install KH Coder on Windows, download and execute the file distributed as the Windows package. Double-clicking the file displays the window shown in Figure A.1. Click the [Unzip] button (without modifying anything) and the .exe file will be unzipped and stored in the folder named khcoder directly under the C: drive. When extraction is complete, a message saying, for example, "5,000 file(s) unzipped

successfully" will be displayed. Click the [OK] button and close this window. The number of files unzipped varies, depending on the version installed.
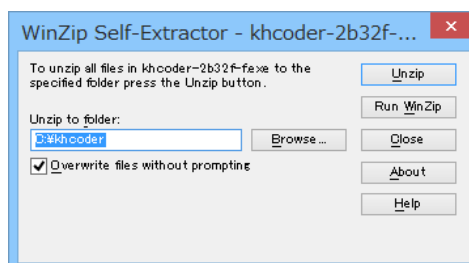
Figure A.1   Set-up (unzipping) of KH Coder on Windows

Executing (double-clicking) the unzipped kh_coder.exe starts KH Coder. Now you are ready to start using the package. For your convenience, add a shortcut to this execution file to the Start menu, or elsewhere, as required. A more detailed description of the above procedure is provided in the Quick Start Tutorial.

On Linux

To install KH Coder on Linux, download the source code file to any folder and unzip it. To start KH Coder, navigate to the folder, where kh_coder.pl is located and run it using the command "perl kh_coder.pl".

A.1.3   Settings

On Windows

The package for Windows performs all configurations automatically, making KH Coder ready for immediate use. No additional manual setup is required; however the following parameters may be modified. Start KH Coder and click [Project] and [Settings] in sequence in the top menu bar. The Global Settings window will now be displayed (Figure A.2).
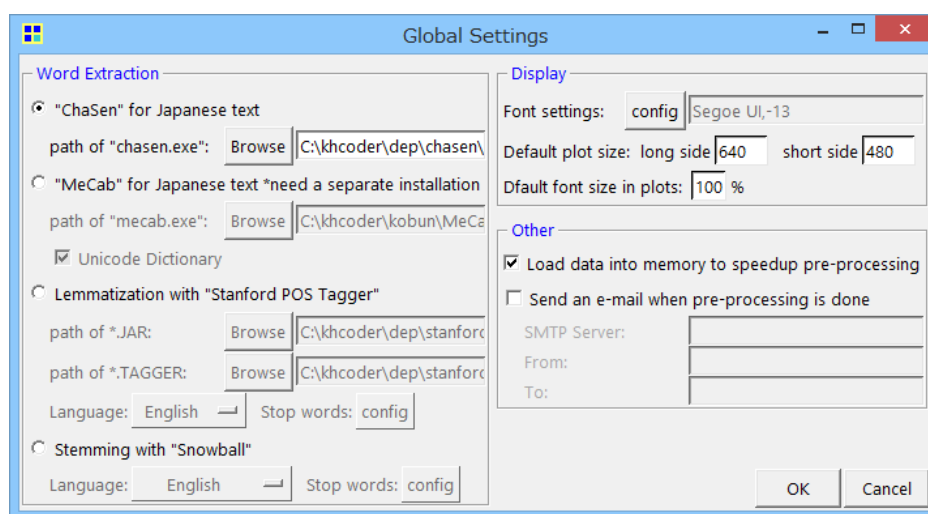
Figure A.2   Global Settings window on Windows

■**Word extraction**   Either ChaSen or MeCab can be selected as the external program for extraction of words from Japanese text data. However, because MeCab is not included in our package for Windows,

it is necessary to install it manually, if you intend to use it. This field allows the specification of the path to the selected "chasen.exe" or "mecab.exe" file. The full path can be entered directly or selected using the [Browse] button.

If MeCab is selected, and the character encoding system employed in its dictionary is Unicode (UTF8), then the encoding system of the text input into MeCab must also be Unicode. To achieve this, check [Unicode Dictionary] under the radio button for the MeCab option. Otherwise, KH Coder inputs Shift-JIS encoded text into MeCab. Even when using the Unicode dictionary, the target file text must be encoded in EUC, JIS or Shift-JIS.

If the target language is English or any other West European language, then select either [Lemmatization. . .] or [Stemming. . .], and specify any stop words. For details, refer to section A.2.4.

■**Manually setting up a connection to MySQL**   KH Coder uses MySQL for data organization and retrieval. Therefore, it is necessary to register all the necessary information concerning MySQL with KH Coder. The package for Windows does this automatically. However, if you intend to modify the settings, refer to the following information.

MySQL-related information must be written manually into a text file named coder.ini in the config folder directly under the folder where KH Coder is installed. The file contains statements, each of which configures one parameter. The statements start with parameter names, such as "sql_username" and "sql_password", followed by tab-separated data fields. To enable KH Coder to access MySQL, at least the user name and password must be set for these two parameters. Additionally, the IP address of the PC where MySQL operates and the connection port may be specified using "sql_host" and "sql_port" parameters, as required.

■**Display features**   Clicking the [config] button of the [Font settings] field displays a window that allows font type and size used in KH Coder windows to be modified. Font settings selected using this window will be used throughout KH Coder, except in menus and plots. KH Coder needs to be restarted to implement such changes.

For plots of statistical analysis results, such as Correspondence Analysis and Co-Occurrence Network, the default plot size and font size also can be specified. These settings can be configured for each analysis window. However, for efficient analysis, we recommend setting optimal values as default values. For example, when a high-resolution display (e.g., a 24-inch, 4 K monitor) is connected, a larger size, such as 1200×900, is more appropriate for plots than the initial 640×480 size. Clearer images can be achieved by setting larger values for both the plot and font sizes in KH Coder with the zoom function in Windows disabled. To disable this zoom function, right-click the kh_coder.exe file, select [Properties] from the menu, and check [Disable display scaling on high DPI settings] via the [Compatibility] tab.

■**Other functions**   To receive an acknowledgement e-mail when pre-processing is complete, check [Send an e-mail when pre-processing is done] and enter user-defined values into the [SMTP Server], [From] and [To] textboxes. The former two settings enable KH Coder to send e-mails and the acknowledgment e-mail will be sent to the address specified in the [To] textbox. Note, however, that this function does not support SMTP servers that require authentication.

Checking [Load data into memory to speedup pre-processing] can somewhat increase the speed of pre-processing (see section A.4.2) with MySQL functions. However, if MySQL has been installed and configured manually, then it is necessary to add the "max_heap_table_size=1024M" statement to the MySQL settings file before using this option. This increases the memory capacity that can be used by MySQL. However, if a data overflow occurs during pre-processing, accompanied by an error message "Failed in accessing MySQL database system.. . . The table 'rowdata' is full", then uncheck this option.

On Linux
Start KH Coder and click [Project] and [Settings] in sequence in the top menu bar. This opens the
Global Settings window (Figure A.2). Given that operation is almost the same as for Windows, only
relevant differences are described below.

■**Word extraction**   Either ChaSen or MeCab can be selected as the external program for extraction
of words from Japanese text data. When using ChaSen, the user must enter the settings file and IPADIC
dictionary file paths to enable KH Coder to access these files. As described later in section A.2.3, KH
Coder will modify these files. Therefore, as recommended in section A.1.1, copying the whole IPADIC
folder into the home folder is more convenient.

■**External applications**   This field allows the settings for external applications to be used in com-
bination with KH Coder. Such settings will be referred to when KH Coder uses external applications
to open output files. Specifically, text strings whose "%s" are replaced with a file name or URL will be
passed to the KH Coder boot shell.

A.2   Basic issues

A.2.1   Data preparation

Limitations
The main limitations of KH Coder when used with Chasen are as follows:

1. All data must be in in plain text format (*.txt). However, Excel or CSV format files can be
   converted into text files using KH Coder (see section A.3.1).
2. The character code for Japanese text data should be Shift-JIS or JIS or EUC. The input data is
   converted to EUC to be processed and accumulated. For this reason, the text data will sometimes
   be garbled, especially when KH Coder finds Unicode characters or model-dependent characters,
   which cannot be converted.
3. In general, when analyzing Japanese text data, we recommended not to use half-width letters in
   your data files. In particular, ',", \, <, >, and | should not be used. However, you can use the
   symbols < and > when they are inserted as HTML header tags for H1 to H5, see next subsection.
   Half-width spaces in the data should be converted to full-width spaces for processing.
4. You cannot use the characters \, <, >, and |, when analyzing alphabetical data. Neither should
   your data contain non-latin1 (non ISO 8859-1) characters.
5. The number of characters in one line or one paragraph without line breaks cannot exceed 4,000
   in full-width. This rule applies to header lines with H1 to H5 tags, as well as normal lines or
   paragraphs.
6. When KH Coder processes a word with more than 126 character width, it will truncate the word
   to 126 characters, and save it. A message will indicate when a word was truncated.
7. You must remove any garbled text, invisible characters, or control characters in your data prior
   to analysis. When analyzing Japanese text data, these can be removed automatically using the
   command "Check the Target File" (see section A.4.1).

Units for coding and searching
■**The purpose of HTML markup**   KH Coder has a function to search "cases" or "units" that meet
specific requirements, as well as a coding function, which gives such "cases" specific codes. KH Coder can
show that words are strongly connected by finding words that often appear together in a given "case".

To use such functions, it is important to make clear, where each case starts and ends. KH Coder enables users to change the "unit level" to indicate where one case starts and ends during your analysis, i.e., dynamically. You can specify "the searching units," "the coding units," and "the counting units" on windows of KH Coder, allowing you to set the units for each analysis type. However, you must have carried out any HTML markup correctly before carrying out such operations.

When you do not need to deal with units other than paragraphs or sentences, you do not need to use HTML header tags. KH Coder delimits a paragraph with a new line and a sentence with a period by default, when it analyzes text files without any HTML header tag markings. This lets you conduct coding, searching, and counting on each paragraph or sentence, as a "case" or "unit". Hence, in the default mode, you can code, search, and count data in paragraph units, as well as sentence units.

When you would like to analyze data on different levels or using an hierarchy, you have to use HTML header tags. You may want to count not only how many sentences or paragraphs deal with specific topics, but also how many chapters or sections include them. You may also wish to distinguish between words appearing in the main text, and words in titles of chapters or sections. To do this, you should insert HTML header tags to create the hierarchy you need in the text file, so that KH Coder can recognize which parts of the text are chapters or sections.

Please note that such header tags are not suitable for expressing multiple types of information or rich information. If you would like to use many variables, like "person," "gender," "age," "date," and "question," it is better to use the variable system in KH Coder. For more information on this topic consult sections A.8.1 and A.3.1 in this manual, or the following slide share tip: http://www.slideshare.net/khcoder/another-example-of-data-preparation-using-variables-not-tags

■**HTML marking for general documents** HTML marking in KH Coder is limited to only five header tags: H1 to H5. KH Coder does not recognize any other HTML tags. You must start a new paragraph immediately after each header tag closure (</h1> to </h5>). In the example below, we show how you can insert tags to a document.

```
1  <H1>The Chapter Title 1</H1>
2  <H2>The Section Title 1</H2>
3  The body of section 1.1 ........................
4  ..............................................
5  <H2>The Section Title 2</H2>
6  The body of section 1.2 ........................
7  ..............................................
8  <H1>The Chapter Title 2</H1>
9  ..............................................
```

You can use either uppercase (H1) or lowercase (h1) tags, because KH Coder treats both in the same manner.

■**HTML marking for newspaper articles** If you want to analyze newspaper articles, you need to carry out coding and searching on each article. You may want to analyze whether articles include specific words or phrases (codes), and whether these increased or decreased through time. To process articles, you will need to mark files as follows:

```
1  <H1>2004</H1>
2  <H2>Jan</H2>
3  <H3>The heading of article 1</H3>
4  The body of article 1 ........................
5  ..............................................
```

```
 6 | <H3>The heading of article 2</H3>
 7 | The body of article 2 ........................
 8 | ............................................
 9 | <H2>Feb</H2>
10 | <H3>The heading of article 3</H3>
11 | The body of article 3 ........................
12 | ............................................
```

■**Coding and searching in each unit**   KH Coder views the HTML header tags hierarchically, with H1 as the highest and H5 as the lowest type of heading. When you select a hierarchical level for analysis, KH Coder ignores all higher text levels and analyzes only the selected level. In the example below (Figure A.3), selecting H3 as the level for analysis requires KH Coder to ignore levels H1 and H2, and analyze only level H3. Effectively, H3 will become the "case." But please note that when you select "sentences" as the analysis unit, a heading with H1 to H5 tag will be treated as a sentence.

```
<H1>2004</H1>
<H2>Jan</H2>
────────────────────────────────────────────────
<H3>The heading of article 1</H3>
The body of article 1 ........................       ⎫
............................................        ⎬ Case 1
────────────────────────────────────────────────   ⎭
<H3>The heading of article 2</H3>
The body of article 2 ........................       ⎫
............................................        ⎬ Case 2
────────────────────────────────────────────────   ⎭
<H2>Feb</H2>
────────────────────────────────────────────────
<H3>The heading of article 3</H3>
The body of article 3 ........................       ⎫
............................................        ⎬ Case 3
────────────────────────────────────────────────   ⎭
```

Figure A.3   Coding with H3 Unit

Although KH Coder basically treats headings with H1 to H5 tags as sentences, as an exception, it will not count a heading whose length is 0 as a sentence. 0 length headings will work as case delimiters, but will not treated as sentences in functions of KH Coder such as coding and searching. For example, when you load an Excel or CSV file as a target file (see section A.3.1), or when you combine multiple text files into one file (see section A.10.2), headings will be automatically inserted to delimit the data. The length of these headings will be 0, so these headings will be non-existent as sentences.

It is best to use heading tags H1 to H5 in the hierarchical manner shown above. For example, for newspaper articles, first divide text into given years with H1 tags. Next subdivide the text into months, using H2 tags. Finally, subdivide each month into smaller parts, or articles, using H3 tags. This results in hierarchical characteristics, where H1 corresponds to a year, H2 to a month, and H3 to an article. Now, you can easily count what percentage of articles cover specific topics, include key words or phrases, and how these attributes change every month or over a number of years. Thus, KH Coder is well suited to dealing with hierarchically-marked data.

If you have to analyze several similar documents, you may only need to delimit them with H1 tags for a simple inter-document comparison. Do not use H2 or H3 tags in these cases, since using different level header tags to delimit same level documents is an incorrect use of tags. It would merely add confusion, without generating any useful information.

A.2.2   Extraction of words

Words with conjugated or inflected forms

When extracting words with conjugated or inflected forms, such as verbs or adjectives, KH Coder converts these words to their base forms and then performs its search. For example, the words "buy", "bought" and "buying" in a given text will all be extracted as "buy"; if all those forms appear at once, then the frequency will be three for the given text.

This processing method reduces the load required for counting word frequency, determining relationships between words, and creating coding rules. In many cases, it may be more convenient to simply specify base forms (e.g., "buy"), rather than conjugated or inflected forms (e.g., "bought" and "buying"), to extract all relevant word forms.

The parts of speech system in KH Coder

The following description applies only when the target language is Japanese and ChaSen is used for word extraction and parts of speech (POS) tagging. If you analyze English data, then "ChaSen" will be replaced by the "Stanford POS tagger".

KH Coder uses the results of the morphological analysis by ChaSen, with almost no modification. Therefore, the POS system complies with that of ChaSen. However, to increase analytical convenience, some modifications and simplifications have been applied. Table A.1 shows the details of the POS system used in KH Coder. "TAG" in the table is the POS name given to strings forcibly extracted as single words (see section A.4.3).

Some Japanese articles and auxiliary verbs that frequently appear in whatever documents are classified as "その他" (other). Words that include no double-byte characters or letters and are extracted as "未知語" (unknown) by ChaSen (e.g., single-byte symbols) will also be classified as this category in the POS system. Such general words are outside the scope of analysis, unless "その他" (other) is checked in the Select Words to Analyze window (see section A.4.3). This is because they are seldom used in analysis. However, the negative auxiliary verbs "ない", "まい", "ぬ" and "ん" are highly likely to be used in coding. Therefore, they are classified as an extra POS, named "否定助動詞" (negative auxiliary verbs).

In addition, nouns, adjectives and verbs that consist of only hiragana will be classified specifically as "名詞 B" (noun B), "形容詞 B" (adjective B), and "動詞 B" (verb B). In this way, we allow them to be excluded collectively from the targets, because hiragana-only words are often too general and are seldom used in analysis, even though they are a major POS.

Modification of the parts of speech system

The following description applies only when the target language is Japanese and ChaSen is used for word extraction and POS tagging. If you analyze English data, then "ChaSen" will be replaced by the "Stanford POS tagger". Likewise, "hinshi_chasen" is replaced by "hinshi_stanford_en."

If you intend to perform linguistic analyses focusing on Japanese grammar or function words, then the default POS settings in KH Coder may not be suitable. In this case, it is best to modify the POS system. When you start KH Coder for the first time, a folder named config appears, containing a file named "hinshi_chasen," which is created directly in the folder where KH Coder is installed. KH Coder loads the POS settings from this file. Therefore, the POS system can be changed by modifying data in this file. Note that any changes made will not apply to existing projects, unless pre-processing is re-executed. The file "hinshi_chasen" is a CSV format file. Its contents are shown in Table A.2. Each row contains a POS name, and its corresponding conditions. All the ChaSen-extracted words will be referred to these rows, from top to bottom in sequence, until they meet the conditions for a particular

Table A.1   Parts of speech system in KH Coder compared with that of ChaSen

POS classification in KH Coder	POS classification in ChaSen
名詞 (noun)	名詞–一般（漢字を含む 2 文字以上の語）
名詞 B (noun B)	名詞–一般（平仮名のみの語）
名詞 C (noun C)	名詞–一般（漢字 1 文字の語）
サ変名詞 (sahen noun)	名詞–サ変接続
形容動詞 (na-adjective)	名詞–形容動詞語幹
固有名詞 (proper noun)	名詞–固有名詞–一般
組織名 (name of an organization)	名詞–固有名詞–組織
人名 (name of an person)	名詞–固有名詞–人名
地名 (name of a place)	名詞–固有名詞–地域
ナイ形容 (nai adjective)	名詞–ナイ形容詞語幹
副詞可能 (adverb)	名詞–副詞可能
未知語 (unknown)	未知語
感動詞 (interjections)	感動詞またはフィラー
タグ (tag)	タグ
動詞 (verb)	動詞–自立（漢字を含む語）
動詞 B (verb B)	動詞–自立（平仮名のみの語）
形容詞 (adjective)	形容詞（漢字を含む語）
形容詞 B (adjective B)	形容詞（平仮名のみの語）
副詞 (adverb)	副詞（漢字を含む語）
副詞 B (adverb B)	副詞（平仮名のみの語）
否定助動詞 (negative auxiliary verbs)	助動詞「ない」「まい」「ぬ」「ん」
形容詞（非自立） (adjective (auxiliary))	形容詞－非自立（「がたい」「つらい」「にくい」等）
その他 (other)	上記以外のもの

row. Their POS names are defined as those in the kh_hinshi column (the remaining rows will be ignored) for that condition.

Each row of in the condition 1 column specifies a string that should match the whole or the first part of the POS name given for words extracted by ChaSen. For example, KH Coder will classify the words, for which the ChaSen POS name is "名詞-一般" (noun - general) or starts with "名詞-一般", with the POS name in the kh_hinshi column of the row, for which the condition 1 string is "名詞-一般".

In the condition 2 column, only "ひらがな (hiragana)", "一文字 (one character)", "HTML" or "否定 (nagation)" can be specified. "ひらがな (hiragana)" indicates that the ChaSen-extracted word should consist of only hiragana, as the corresponding POS name implies. Similarly, "一文字 (one character)" or "HTML" indicates that the word should consist of a single character or should be an HTML tag from H1 to H5, respectively. Lastly, "否定 (negation)" means that the base form of the word is "ない", "まい", "ぬ" or "ん" (negative auxiliary verbs). If any of those four parameters is specified in the condition 2 column, then the word should meet both condition 1 and condition 2 to be classified with the corresponding POS name in KH Coder.

Additionally, if you want to create a new POS name, then enter it into the hinshi_id column as a unique number, i.e., one that is not already in use for other POS names. As for "感動詞" (interjections), more than one conditional entry can be made for a POS, so as to classify words with different ChaSen POS names, as the same POS in KH Coder.

A.2.3   Configuration of ChaSen and MeCab

KH Coder modifies the settings of ChaSen automatically, as required. If you intend to change the settings manually, be sure your changes do not conflict or interfere with ones made automatically by KH Coder.

Table A.2 Contents of the parts of speech settings file (config/hinshi_chasen)

hinshi_id	kh_hinshi	condition 1	condition 2
7	地名	名詞–固有名詞–地域	
6	人名	名詞--固有名詞--人名	
5	組織名	名詞--固有名詞--組織	
4	固有名詞	名詞--固有名詞	
2	サ変名詞	名詞--サ変接続	
3	形容動詞	名詞--形容動詞語幹	
8	ナイ形容	名詞--ナイ形容詞語幹	
16	名詞 B	名詞--一般	ひらがな
20	名詞 C	名詞--一般	一文字
21	否定助動詞	助動詞	否定
1	名詞	名詞--一般	
9	副詞可能	名詞--副詞可能	
10	未知語	未知語	
12	感動詞	感動詞	
12	感動詞	フィラー	
99999	HTML タグ	タグ	HTML
11	タグ	タグ	
17	動詞 B	動詞--自立	ひらがな
13	動詞	動詞--自立	
22	形容詞（非自立）	形容詞--非自立	
18	形容詞 B	形容詞	ひらがな
14	形容詞	形容詞	
19	副詞 B	副詞	ひらがな
15	副詞	副詞	

Table A.3 Content of the "hinshi_stanford_en" setting file

hinshi_id	kh_hinshi	condition 1	condition 2
2	ProperNoun	NNP	
1	Noun	NN	
3	Foreign	FW	
20	PRP	PRP	
25	Adj	JJ	
30	Adv	RB	
35	Verb	VB	
40	W	W	
99999	HTML_TAG	TAG	HTML
11	TAG	TAG	

Automatic changes made by KH Coder to settings are described below.

Changes to the grammar.cha file (in ChaSen)

Regardless of the settings in KH Coder, the following two statements will be added to the "grammar.cha" file in ChaSen.

```
1  （複合名詞）
2  （タグ）
```

Changes to the chasenrc file (in ChaSen)
Regardless of the settings in KH Coder, the following two statements will be added to the "chasenrc" file in ChaSen. However, the addition of the "文法ファイル" (grammar file) statement occurs only to the Windows version of KH Coder, and its contents will vary depending on where KH Coder and ChaSen are installed.

```
1  (文法ファイル  "C:/khcoder/dep/chasen/dic")
2  (注釈（("<" ">")（タグ)))
```

Finally, if the [use ChaSen] command under [Word Clusters] (section A.4.5) is executed, then the following statements are added to the "chasenrc" file temporarily.

```
1  (連結品詞
2      ((複合名詞)
3          (名詞)
4          (接頭詞名詞接続)
5          (接頭詞数接続)
6          (記号 一般)
7      )
8  )
```

These statements will be deleted when the command is completed.

Settings for MeCab
KH Coder will not modify any MeCab settings. If you intend to employ a user-defined dictionary in MeCab for analysis, then it is necessary to add a statement concerning its use to the "mecabrc" file. KH Coder for Windows refers to the "mecabrc" file as a settings file, located in the folder with mecab.exe. Thus, if the required files like "mecab.exe" and "mecabrc" exist, then MeCab can be used without installing it.

Exchanging dictionaries
If a non-default (non-IPADIC) dictionary of ChaSen or MeCab is used, then you need to confirm the POS system or POS names used in that dictionary in advance. If the POS names are different from those in IPADIC, then it is necessary to modify the POS settings in KH Coder (see section A.2.2). In the default settings, KH Coder processes words, for which the IPADIC POS name is "名詞-一般" (noun - general) as nouns. However, in non-IPADIC dictionaries, such words may be registered under a different POS name, for example, "名詞-普通名詞-一般" (noun - common noun - general). In such cases, the POS settings in KH Coder must be changed from "名詞-一般" to "名詞-普通名詞-一般".

However, just changing the POS settings is not sufficient to allow the use of [Word Clusters] commands. To use these commands, it is also necessary to rewrite their processing details. The KH Coder packages provide command modification samples as plugins for when UniDic is used instead of IPADIC. Refer to these plugins, as required (see section A.10.6).

A.2.4   Data in English and other non–Japanese Languages

Lemmatization and stemming
You can choose either lemmatization, which uses the "Stanford POS Tagger," or stemming, which uses the "Snowball Stemmer," when you analyze English data with KH Coder (section A.2.4). Stemming is a process that cuts ends of words, according to simple rules. For example, it can extract "says" or

"saying" as the original form "say," but extracts "said" as just "said." In contrast, lemmatization can also extract "said" as "say" by using the built-in dictionary.

In KH Coder, stemming does not classify words using the POS system. For example, it extracts both the gerund "thinking" and the present particle "thinking" as "think." It treats both words as the same. In contrast, lemmatization does try to distinguish between the gerund form "thinking" and the verb "think".

Before you decide which one to use, consider the advantages and the disadvantages stated below. Clearly, your choice will also depend on the purpose of your analysis. The advantages of lemmatization are that it can extract words based on linguistics, and you can choose words based on their POS classification when you analyze your data. The main disadvantage is that it takes a long time to process data. If you cannot decide which method is more appropriate, we recommend that you try lemmatization first. The advantages of stemming are that it has a faster processing time and its extraction is accurate enough to analyze semantic content. Its main disadvantage is that its extraction is less linguistically accurate than lemmatization. However, if you do not need to distinguish between the gerund verb form "thinking" and the verb "think," then stemming will be suitable for your purposes.

You can modify the POS system by changing the configuration file, as described above for Japanese text data. The name of the configuration file for lemmatization (English) is "hinshi_stanford_en", while the one for stemming is "hinshi_stemming_en". If you use lemmatization, the Stanford POS Tagger outputs the names of various POS based on the Penn Treebank tag set. Under the default setting, KH Coder divides data into the simplest POS, such as verbs and nouns, although the tag set includes conjugated forms as distinct entities. However, KH Coder also saves the original POS classification of the Penn Treebank, as "conjugation." Hence, you can search and count data, based on this classification in KH Coder.

Handling stop words

In general, common words occurring in all kinds of sentences, such as substantive verbs, are specified as "stop words" and are removed from analysis. However, KH Coder has no default list of stop words, so you need to set it according to your analytical purpose. We recommend that you select words based on the sample list of stop words in the Quick Start Tutorial for KH Coder.

Stop words set using the settings window (Section A.1.3) are categorized as "other" in the POS system. KH Coder evaluates whether or not a word is a stop word, only after it has processed the text using lemmatization or stemming. Therefore, it is only necessary to input the original form of a word, like "say" rather than "saying", when you set stop words.

Words set as stop words are omitted from analysis because "other" POS are exempt from analysis. However, they are included when counting the number of words in a document. Although, if you specify words as "force ignore" on the settings window under "selection of words", then they are handled as nonexistent (section A.4.3).

Other European and Asian languages

The latest version of the KH Coder can analyze data in Chinese (simplified), French, German, Italian, Korean, Portuguese, Russian and Spanish. you need to set the character encoding for these data as UTF-8 or Latin1 (ISO 8859-1).

Analysis functions for these languages are still in the preliminary stage, and sufficient testing has not yet been carried out. Therefore, we recommend that you check results of such analyses carefully. If you do find errors or points to be improved, I would appreciate your feedback.

Details of each analysis step

Unlike for Japanese text data, you cannot simply punctuate a sentence in English or other Western European languages. For this reason, sentence splitting should be dependent upon language. KH Coder uses a Perl module "Lingua::Sentence" for this purpose. This process is common to both lemmatization and stemming.

The next step involves tokenization. To pick out words from sentences, stemming follows the original process in KH Coder, while lemmatization uses the Standard POS Tagger. In KH Coder, the Penn Treebank protocol is used for tokenizing English data. Thus, words are divided according to English rules, for example, "aren't" is separated into "are" and "n't". However, for other languages, KH Coder can only conduct a simple process, separating out quotation marks or periods at the end of a sentence. In addition, it can separate the beginning of a French word, starting with c', d', l', and s', but it still cannot apply such rules to other languages. If you are aware of any documents on standard tokenization in any of these other languages, available in Japanese or English, please let me know, so that this feature can be further improved.

Lastly, KH Coder uses the Standard POS Tagger to lemmatize separated words, and the Snowball Stemmer for stemming. The Snowball Stemmer follows a set of rules in each language. Hence, KH Coder can analyze languages supported by both the Snowball Stemmer and the "Linugua::Sentence" module. However, at this preliminary stage, there is limited support for Western European languages.

A.2.5 Coding rules

How to make a coding rule file

If you write rules for coding as a protocol file, KH Coder can automatically carry out coding according to these rules. You can create simple protocols, such as any document including either "arrest" or "suspect" or "investigation" is given the code "crime"; but also more complicated ones that combine multiple conditions, using AND, OR, and NOT logical operators.

KH Coder can make cross tabulation tables for coding results, as well as simply counting codes. Likewise, it can estimate relationships between codes, make a list of words that have a strong relationship with a specific code, and search only documents to which that particular code applies. Coding rules are useful not only for counting, but also for searching, since we can combine various search conditions into coding rules.

One text document can comply with multiple coding rules, depending on your rules. In such case, all codes are applied to this document. KH Coder does not categorize a document into exclusive categories, such as "legal" or "crime", using coding rules. Rather, it extracts such elements from a document. Thus, the coding in KH Coder is based on the premise that one document can contain multiple elements, such as "crime" and "warmhearted".

KH Coder can read coding rules from a text file. Therefore, you can write and edit protocols for KH Coder using a simple text editor, such as Notepad. The format for writing coding rules is given below:

```
1  *Name_of_a_Code_1
2  Condition to Apply to Code 1
3  # Memo about Code 1 (Write only if required)
4
5  *Name_of_a_Code_2
6  Condition 2
7
8  *Name_of_a_Code_3
```

```
9 │ Condition 3
```

Before each code name, an "*" (asterisk) should be added, so that KH Coder recognizes it as the name
of a code. In the next line, you need to specify the condition for that code, and in what kind of situation
it should be applied. Using this format, you can write as many codes as you need for a given text file.

Because KH Coder simply ignores blank lines, you can structure a coding rule file using blank lines
so that it is easier to read. It also ignores any line where the first character is "#"; hence, these can be
used as comment lines. We recommend that you provide explanations or memos on codes as comments
for future reference.

Rules can written as Boolean expressions or logical expressions with delimiters such as +, -, ==, !, &,
|, and ( ). Further details are provided below.

Writing conditions
■**A single word**   The simplest condition is to specify a single word. When you specify one word as
a condition, you should write the word in its base form or use the lemma of the word. For example, to
apply a code "*Crime" to documents containing the word "arrest", you can write the coding rule, as
shown below.

```
1 │ *Crime
2 │ arrest
```

If you would like to specify a POS classification as well as words, you write "word-->POS", without a
space between the word and the symbol. Do not insert spaces in this type of expression. Please note that
you do need to insert spaces before and after symbols, when you use arithmetic and logical operators,
as will be discussed later in this section. For example, if you want to apply a code "Sound_Verb" to the
verb "sound", e.g. "It sounds funny", as well as a code "Sound_Noun" to the noun "sound", e.g. "I
heard a splashing sound", you can write these coding rules, as follows:

```
1 │ *Sound_Verb
2 │ sound-->Verb
3 │
4 │ *Sound_Noun
5 │ sound-->Noun
```

If you simply specify "sound", without listing its POS classification, then all of the words related to
"sound" are targeted, regardless of their POS classification.

Likewise, if you would like to provide a code for a word with a specific conjugation, you can write
"word->conjugation", without any spaces between the word and symbol. For example, if you wanted to
specify the form "sounded", then you can write this coding rule, as follows:

```
1 │ *Sounded1
2 │ sound->VBD
```

Without specifying a particular conjugation, "sound" (VB), "sounds" (VBZ), and "sounded" (VBD)
are all targeted by the same coding rule. If you want to specify both a POS classification and a conjugation
simultaneously, you can write: "word-->POS->conjugation".

If you would like to specify a conjugate word, without reference to its conjugation, such as "VBD" for
the word "sounded", you can write this rule, as follows:

```
1  *Sounded2
2  sound=>sounded
```

It also is possible to write a coding rule, such as "word-->POS=>surface_form" in this case. However, you cannot specify both the conjugated word and the name of its conjugation in the same rule. In this case, KH Coder distinguishes between uppercase and lowercase letters in the specified surface form as an exception, although specifying either uppercase or lowercase will lead to the same result in assigning a basic form. This exception has been implemented because when you specify not only the lemma form of the word, but also its conjugated form, it is likely that you will want to differentiate lowercase and uppercase as well.

■**A single phrase**   When you want to specify a phrase, rather than a single word, you should write "Word1+Word2+Word3+...". For example, if you want to provide a code "*Graduation_thesis" for a phrase including "graduation" and "thesis" successively, you can write the protocol as follows:

```
1  *Graduation_thesis
2  graduation+thesis
```

In this case, there are no spaces between words and the "+" symbol.

■**Nearby words**   You may like to define a code that has specific words in nearby context. To do this, you can write "near (Word1-Word2-Word3...)". For example, the code "*Alice_and_Ken_Couple" below defines the case where "Ken" appears close to "Alice".

```
1  *Alice_and_Ken_Couple
2  near(Ken-Alice)
```

Without recourse to additional options, words are recognized as "close", if they appear within 10 words of each other, either before or after. In the case above, a code "*Alice_and_Ken_Couple" is given when "Alice" exists within 10 words of "Ken". If you want to change it to within 3 words, you can add a [3] to this condition, as in "near(Ken-Alice)[3]". You also can specify more complicated conditions by using the letters, "b", and "d", as shown in examples below.

- [15] Within 15 words of each other
- [b] In the same sentence AND within 10 words
- [b15] In the same sentence AND within 15 words
- [b0] In the same sentence, no matter how far apart
- [d] In the same paragraph AND within 10 words
- [d15] In the same paragraph AND within 15 words
- [d0] In the same paragraph, no matter how far apart

As you can see from this list, the coding rule for "(before and after) within 10 words" is automatically added, without assigning numbers. If you write "0", KH Coder skips checking how far apart specified words are in a given text. Hence, if you write "near(Ken-Alice)[0]", KH Coder will apply the code to all documents containing both "Ken" and "Alice". However, this type of general specification is not recommended; it is better to write "Ken & Alice" using the logical operator "&", as described later in this section, since this format is easier to read and shortens processing time.

The use of "b" and "d" are ignored, when coding and searching is carried out sentence by sentence. Similarly, for coding and searching every paragraph, "b" works, but "d" is disabled. Please refer to section A.2.1 for information on units for coding and searching. The current version of KH Coder does

not allow one to use the same word twice in this type of condition. For example, you cannot write a condition for "two Ken and one Alice are close", nor expressions like "near(Ken-Ken-Alice)".

■**Word order**   In KH Coder you can specify a coding rule dealing with word order, e.g., "After one word appears, the others appear in sequence", by writing "seq(word1-word2-word3...)". If you write a rule like the example given below, then a code "*Ken_after_Alice" is applied, when "Ken" appears close to, but after "Alice".

```
1  *Ken after Alice
2  seq(Alice-Ken)
```

The condition of proximity for which the code is defined is the same as for "Nearby words", shown above. You can also add the same options for both these specifications. KH Coder processes such specifications for "Nearby words" and "Word order", basically in the same way. The only difference is that word order is not considered in the conditions for "Nearby words".

Although a [0] is not a helpful option in "Nearby words", it can be useful when analyzing word order, depending on your condition. If you write "seq(Alice-Ken)[0]", then a code will be given wherever "Ken" appears after "Alice", without consideration of how close they occur. While, if you add [1], then situations only when the word is immediately after the other are recognized as "close". Therefore, "seq(graduation-thesis-defense)[1]" is synonymous with "graduation+thesis+defense", in the phrase search expression defined above.

■**Arithmetic operators**   When you want to give a code based not on whether the word appears, but how often the word occurs in a given text, you can use arithmetic operators to define conditions, as listed in Table A.4. For example, you can write a coding rule, such as "*Mail_twice" to be applied to documents, where the word "mail" appears twice. You can write this rule as shown below.

```
1  *Mail_twice
2  mail == 2
```

Table A.4   Arithmetic operators in KH Coder

Operators	Meanings
+	addition
−	subtraction
∗	multiply
/	division
>	greater than
<	smaller than
>=	greater than or equal to
<=	smaller than or equal to
==	equal to

You need to insert a space before and after an operator, when you use arithmetic or logical operators. If no spaces are included, then an error message will be displayed, and the coding will not be performed correctly.

You can also prescribe a code such as "*Mail_multiple_times" to identify text where either mail or message appears twice or more. This is written as:

```
1  *Mail_multiple_times
2  mail + message >= 2
```

In the process of coding with arithmetic operators, extracted words are replaced by the number of times they appear, and then KH Coder determines whether the arithmetic expression is satisfied or not. Here, the number of times the word appears is determined for each specific document, so this check is repeated as many as there are documents. Therefore, it is useful to specify a condition, such as "mail + message > website", so that you can compare the number of words within each of the documents.

■**External variables** If you load external variables (section A.8.1), you can specify conditions based on their values by writing "<>name_of_variable->value". Here, no spaces are included between a symbol and the name of a variable, or between a symbol and the threshold value. This type of rule enables us to designate codes, such as "*Answer_by_male" to identify answers by males in open-ended questions in questionnaires. This coding rule can be defined as:

```
1  *Answer by male
2  <>gender-->male
```

This example assumes that the variable "gender" is loaded externally and its values are "male" and "female". You can also use the value of a variable itself, as well as its label (section A.8.1) for coding rules. If the names of variables or their values contain any spaces, then you need to put them in double quotations, such as in the example "<>Chapter-->Chapter I". If the value has a double quotation ("), you can specify this double quotation by writing it twice, consecutively (""). However, we do not recommend using spaces or double quotations in the names or values of variables.

■**Headings** You can also use the special external variables "Heading1" to "Heading5", if you have marked the text using HTML tags (H1 to H5). For example, for the text with HTML marking shown in section A.2.1, you can write a rule, such as:

```
1  *Documents Included in the First Chapter
2  "<>Heading1->The Chapter Title 1"
```

The number of headings will correspond to their tags, where "Heading1" is applied to headings enclosed in H1 tags, and "Heading2" is applied to headings enclosed in H2 tags, and so on.

■**Sequential ID number of document** You can define a number that shows where the document is located with respect to the beginning of the text file by writing "No.", as shown in the next example.

```
1  *The_Fifth_Document
2  No. == 5
3
4  *The_Documents_after_the_99th
5  No. > 99
```

For example, if you have found any unique characteristics for a specific document using multivariate analysis or automatic classification, this type of coding rule enables you to search and view this document quickly.

■**Length of a document** You can specify a condition to determine the length of a document by using the special character strings, "lw" and "lc". The following example gives the protocol to apply a code only to a document containing less than or equal to 10 words.

```
1  *Documents_contain_Less_than_or_Equal_to_10_Words
2  lw <= 10
```

This example shows that the number of words in each document is counted, when you type "lw". In contrast, "lc" is used to count the number of characters in each document.

■**Strings**   When you would rather specify conditions using strings in the text file, rather than words, you should enclose them in single quotes ('). For example, you can define the following protocol to identify text with the string 'are you sure'.

```
1  *Confirmation
2  'are you sure'
```

In most cases, it is more useful and faster to specify conditions using words extracted by KH Coder to perform both coding and searching. However, if you skip the procedure to extract a word forcibly (section A.4.3), then this type of specification is effective for cases where you would like to search for words that KH Coder did not extract automatically. Please note that you cannot specify strings separated into multiple sentences.

Combining multiple conditions

We have defined how to specify conditions ranging from whether or not a word appears, the number of times a word appears, the length of a document, and how to identify external variables in a text. In this part, we focus on ways to combine these conditions, using logical operations. Table A.5 shows logical operators that can be used for coding rules. Their meanings and examples are outlined in more detail below. As for using arithmetic operators, it is necessary to insert spaces before and after logical operators, as well as parentheses to ensure coding is performed correctly. Although in the following examples we use symbols, such as "|" and "&", it is possible to replace these with "or" and "and", listed as their alternatives in Table A.5. If you are handling English text data, and would like to specify "and", "or", or "not" as an extracted word, you can write these by enclosing them in double quotes, as in the example shown here: ""and"".

Table A.5   Logical Operators available in KH Coder

Operators	Alternative
|	or
&	and
!	not
&!	and not
|!	or not
( )	*None Available*

■**"|" Operator**   It is possible to specify multiple conditions using "or" conjunctions. For instance, it is possible to designate a condition, such as "When either "arrest" or "suspicion" or "investigation" exits in a document, then a code "*Crime1" is applied to it". You can write this rule using logical operators, as follows:

```
1  *Crime1
2  arrest | suspicion | investigation
```

■**"&" Operator**   This is used to specify multiple conditions with "and" conjunctions. For instance, it is possible to designate a condition, such as "When "communication", "interception", and "bill" appear in a document together, a code "*Wiretapping_Law" is applied". This coding rule can be written as:

```
1  *Wiretapping_Law
2  communication & interception & bill
```

■**"!"   Operator**   This operator is used to exclude or deny a condition. You can only use it at the beginning of a condition description; hence, you should use "&!" or "|!" for more complex cases. For example, you can specify a condition, such as "A code "*Innocent" is given, if the document does NOT contain the word "guilty"", as follows:

```
1  *Innocent
2  ! guilty
```

■**"|!"   Operator**   You can use these operators when you need to specify a condition with an "or not" conjunction. You can define a condition, such as "Either when a word "escape" exists or when a word "arrest" does NOT exist, then a code "*Runaway" is provided", as outlined below.

```
1  *Runaway
2  escape |! arrest
```

■**"&!"   Operator**   These operators can be used together for conditions that involve an "and not" conjunction. You can specify a condition, such as "If a word "wiretap" appears AND a word "bill" does not appear in the document, then a code "*Criminal_Wiretapping" is applied". This rule is written, as follows:

```
1  *Criminal_Wiretapping
2  wiretap &! bill
```

■**Parentheses**   You can combine the operators above using parentheses to define more complex conditions. In the example below, a code *Crime2 is applied when all three words ("communication", "interception", and "bill") do NOT appear together AND either "arrest" or "suspicion" or "investigation" appears in the same document.

```
1  *Crime2
2  ( arrest | suspicion | investigation ) &! ( communication & interception & bill )
```

Parentheses can be nested to define complicated conditions. Although the example above specifies the condition based on whether or not the words exist, you can also combine other type of conditions, such as external variable values or the length of the document into an argument.

■**Use of predefined codes**   Using logical operators, you can reuse codes you have already defined in the same coding rule file. You only need to enclose their code name in square parentheses (<>) to do so. For example, if you write a protocol for a code called "Education" which is given to documents that contain the words "school", "teacher", "tutor", etc., then a code "Interaction_in_Education" is applied only to documents that contain the word "interaction" too (see coding example below).

```
1  *Education
2  school | teacher | tutor | ... (the rest is omitted)
3
4  *Interaction_in_Education
```

```
5  <*Education> & interaction
```

Notes on writing conditions

■Using spaces   You need to insert spaces before and after operators, as well as parentheses, when you use either arithmetic or logical operators, i.e., you must place a space between a word and an operator, a word and a parenthesis, and an operator and a parenthesis.

However, you should be careful not to insert any spaces before and after marks such as "-->", which are used to specify a POS or a conjugation. You also must make sure not to insert any spaces before and after the parenthesis used in "near" and "seq" conditions. In short, it is necessary to put a space before and after a parenthesis in arithmetic and logical operator expressions, but it is prohibited to insert spaces before and after any other notation marks.

You must use ASCII characters to write arithmetic and logical operators, as well as all notation marks like "-->", "->" and "<>". You cannot use double-byte or full-width characters for these symbols.

■Using Tabs and Line Breaks   You can use line breaks, tab characters, and spaces freely to write readable coding rules. For example, the following rule is difficult to interpret and read without any formatting:

```
1  *Bulletin Board System (BBS)
2  bulletin+board | (BBS &| ( ( Big & Brothers & and
3  & Sisters ) | ( Big & ( Brother | Brothers ) & and
4  & Sisters ) | ( aluminum & wheel ) | ( England &
5  program) | BigBrothersandSisters ) )
```

This same protocol is read more easily, when structured:

```
1   *Bulletin Board System (BBS)
2   bulletin+board
3   | (
4       BBS
5       &! (
6             ( Big & Brothers & and & Sisters )
7           | ( Big & ( Brother | Brothers ) & and & Sisters )
8           | ( aluminum & wheel )
9           | ( England & program)
10          | BigBrothersandSisters )
11      )
12  )
```

A.2.6   Window operations

When you start KH Coder, the window shown in Figure A.4 opens. This is the main interaction window for KH Coder. Even when other tool windows are open, KH Coder exits when you close this main window. You can close windows in KH Coder using ESC or Ctrl + Q, as well as clicking the [X] in the upper right corner of the window. However, unlike other windows, the main window cannot be closed using the ESC key. This shortcut setting prevents accidental termination of KH Coder. If you press Alt +M, while other tool windows are active, then the main window becomes active.

At the "Interface Language" tab in the lower right corner of the window, you can select either "Japanese", "Chinese", "English", "Korean" or "Spanish". You can switch the language used for menus
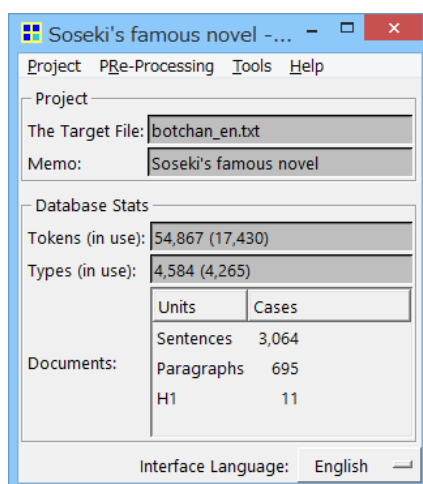
Figure A.4   Main interaction window in KH Coder

and buttons in windows of KH Coder using this option. However, this only changes the display language, and functions in KH Coder will not change. Therefore, if you want to change the language of the target text file from Japanese to English, you must select the language when you create a new project (see section A.3.1).

"Tokens" displayed in the main window indicate the total number of words in the target text file, while "Types" gives the number of types of these words. Please note that HTML tags H1 to H5 are not counted. A numerical value in parenthesis indicates the number of words KH Coder recognizes as analysis targets. Without changing any settings, common words, including definite or indefinite articles and auxiliary verbs, which exist in whatever documents, are excluded from analysis (see section A.2.2). These words also are excluded from totals displayed in parentheses under "Tokens" and "Types"that appear once pre-processing is complete (see section A.4.2).
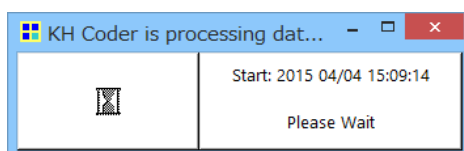

Figure A.5   Display during processing

When the size of the target file is large, more than tens of minutes are required for some processes, including pre-processing (see section A.4.2). When you are running a process, which requires more than a few minutes, KH Coder displays a dialogue box to determine whether to execute this process immediately. Therefore, you can temporarily suspend the execution of this process, if you would like to carry out other tasks beforehand. Figure A.5 shows the window that will be displayed during processing.

During processing, KH Coder does not maintain or update the screen. Hence, in some cases, KH Coder windows corrupt or else OS (Windows) recognizes it as having "no response", despite its normal execution. However, if neither the HDD nor the CPU works, in combination with a "no response" message or a corrupt window, then you need to suspect the possibility that some error has occurred. In such cases, we recommend using the console window to check whether there are any error messages, as well as the operating conditions of the HDD and the CPU. The console window is normally minimized on Windows, displaying technical information in white characters on a black background.

A.3   [Project] menu

A.3.1   Create a [New] Project

Overview

To analyze data, the target file has to be registered in KH Coder as a "project" using this command. Executing this command opens a New Project window (Figure A.6). In this window, you can specify the target file, the language of the data and the word extraction software. Currently, you can analyze Japanese, English, Chinese, French, German, Italian, Korean, Portuguese, Russian and Spanish language data with KH Coder. When you analyze English language data, you can select "Stanford POS Tagger", "FreeLing" or "SNOWBALL" as the word extraction software. After you selecte the file and the language, add a description (memo), as needed and click the [OK] button to create the project.
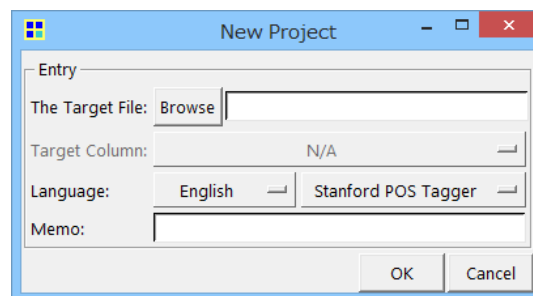

Figure A.6   Create a new project

This registration operation needs to be carried out only once. To start analysis of a registered file, use the "Project" > "Open" command defined below (section A.3.2).

To date, KH Coder has no "Save" command to save a project. Results of pre-processing (section A.4.2) will be automatically saved to the MySQL database. Therefore, you only need to be mindful to save the coding rules file, and any important results of your analysis. There is no need to save a project.

CSV and Excel format files

Figure A.7 provides an example in which the target text is saved as an Excel worksheet. External variables (see section A.8.1) are contained within the same worksheet. The following two preparation steps are required to process these data. Firstly, save the target text data as a text format file (*.txt), and save the variables, separately, as an external variable file. Secondly, register both files with KH Coder. To facilitate this procedure, we have incorporated an automatic conversion function in KH Coder for CSV and Excel files. Hence, if a CSV or Excel file is specified as the target file, then KH Coder automatically converts it into two text format files for target text data and variable data. It also registers the text data file as a project, and imports the variable data file as external data.

To use this automatic conversion function, files must be prepared with the format shown in Figure A.7. Specifically, the first row must be headings that contain column names relevant to their contents. The data to be imported must be in the first sheet of the file. If the Excel file shown in the A.7 is specified as a new project, then "text", "chapter", "other_variable1" and "other_variable2" are all made selectable for [Target Column:] in the New Project window (Figure A.6). Select one of these three columns as the text data to be analyzed, and click [OK].

Two new files will now be created in the folder, where the original Excel file was located. One file has a name consisting of the original Excel file name followed by "_txt0.txt"; it contains the data from the column specified in [Target Column:]. This file is registered with KH Coder as the target file. Be advised

	A	B	C	D
1	**text**	**chapter**	**other_variable1**	**other_variable2**
2	Because of an hereditary recklessness, I have been playing a	I	a	1
3	"I'll see I don't get dislocated next time," I answered.	I	b	2
4	One of my relatives once presented me with a pen-knife. I w	I	c	3
5	"Rather dull? See if they don't cut!" I retorted.	I	a	4
6	"Cut your finger, then," he challenged. And with "Finger noth	I	b	5
7	About twenty steps to the east edge of our garden, there wa	I	c	1
8	One certain evening I hid myself behind a folding-gate of th	I	a	2
9	Besides the above, I did many other mischiefs. With Kaneko	I	b	3
10	Father did not like me in the least, and mother always sided	I	c	4
11	"This fellow will never amount to much," father used to rem	I	a	5
12	"He's so reckless that I worry about his future," I often heard	I	b	1

Figure A.7   Example of an Excel file suitable for automatic conversion

that the project will fail to open, if this file is moved or deleted. Inside this file, each cell is separated by a heading enclosed with H5 tags: "<h5>---cell---</h5>". During analysis and search functions, selecting "H5" (heading 5) as the tabulation unit allows you to search by cell. In addition, these headings prevent any problems derived from line changes in cells, because they explicitly demark the start and end of each cell. Line changes within cells are retained in the file, as they were. The "---cell---" string is configured to be extracted forcibly as a single word, and then ignored. Therefore, they will not be counted as words in any text analysis. In practice, they are treated, as if they did not exist. These settings can be confirmed in the Select Words to Analyze window (see section A.4.3).

The other file has a name consisting of the original Excel file name followed by "_var0.txt" and contains the data from the columns not selected as the target. This file is imported automatically as external variables. If a file with the same name already exists, a new file will be created by incrementing the number digit in the file name.

Internal processing
When a new project is created, a dedicated MySQL database is created for saving all project data. Project information, such as the name of the target file, the description (memo), and the name of MySQL database is saved in a CSV file "config/projects" in the folder, wherever KH Coder was installed.

A.3.2 [Open] an existing project

Overview
This command is used to open projects already registered using the [New] project command (section A.3.1) to continue with an analysis. Executing this command opens the Project Manager window (Figure A.8). Select the project you want to continue analyzing from the list shown, and then click [Open].

Options
The Project Manager window allows users to delete unnecessary projects using the [delete] button. However, this procedure only removes its registration information from KH Coder. It does not delete the target file. A currently open project cannot be deleted. Instead, the project needs to be closed before deleting it, using the [Project] > [Close] command.

Clicking [edit] opens a window similar to that shown in A.6. This window allows the user to make changes to the description of existing projects or the language setting of the target file. In addition, a new project can be created by clicking on the [New] button in this window (section A.3.1).
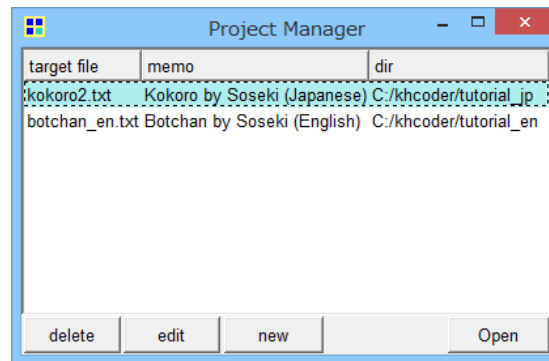
Figure A.8   Project Manager window showing registered projects

A.3.3   [Import and Export] projects

Overview

Using the [Export] command, all data in an existing project can be written to one file. This exported file can be imported into KH Coder as a project, using the [Import] command. Thus, the exported file can be copied and imported onto another computer to continue with analyses. If these commands are not used to analyze the same data on other computers, then the series of operations involved in creating a new project, i.e., pre-processing, and specifying various settings, will need to be carried out again. The [Import] and [Export] commands simplify this procedure.

Internal processing

In the export process, all data in the MySQL database are written out to one text file. In addition to this file, the target file and a file with the project information are compressed into a zip file, and saved with the *.khc filename extension.

These operations require a certain amount of processing time. In particular, the import process can take long because it involves the reconstruction of a MySQL database based on the text file. The exported file is at least ten times larger than the original target file.

A.3.4   [Settings]

Overview

This command enables you to define global settings, common to all projects. For details, please refer to section A.1.3.

Internal processing

Global settings are saved to a "config/coder.ini" file in the folder where KH Coder was installed. In addition to the global settings, the position and size of each window is saved to this file.

A.4   [Pre–Processing] menu

A.4.1   [Check the Target File]

Overview

Currently, this command is available for Japanese text data only. This command allows automatic data correction, such as deleting of garbled characters (Mojibake) and wrapping of long lines in the target file.

Especially, when data is collected online, it may contain Mojibake or lines that are too long. Analyzing such a file may cause ChaSen to terminate abnormally, before completing its processing. Thus, the target file should be corrected using this command before starting analyses. For details on the restrictions on a target file, please refer to section A.2.1. This command also displays a warning, when H1 to H5 tags are (or assumed to be) misused, or when headings defined with H1 to H5 tags are too long. These issues must be manually corrected. If any problems are identified in a target file through executing this command, then the window shown in Figure A.9 will be displayed.
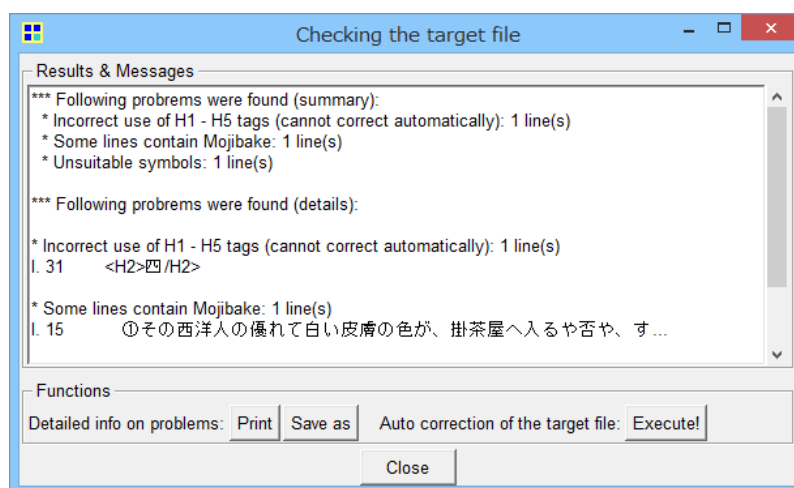
Figure A.9   Window showing the results of checking the target file

Options

■**Details of Results**   At first, only summary information on any problems identified is shown in the window. Clicking the [Print] button, displays more detailed information.

As shown in Figure A.9, this window displays the line number of the line that contains a problem, as well as the first 60 characters of this line. Clicking on the [Save as] button, saves the window content, as a text file. It is important to be aware that even though a line wrap is performed with the automatic correction, it still displays the line number before any correction was applied. Therefore, you need to be careful with the differences between the displayed line number and the actual line number. To accurately identify the line number of a line that contains a specific problem, which is not automatically corrected, you may need to close the window shown in Figure A.9, and re-execute this command.

■**Automatic correction**   Clicking the [Execute] button shown in Figure A.9 performs an automatic correction. As the automatic correction is carried out, a backup file is automatically created, and all corrected content is saved as a differential file. The names of these backup and differential files are displayed in the window. Generally, a backup file is saved with a file name, such as "coder_data/[target file name]_bak0.txt." The content of the differential file is the output of the GNU diff command from freeware GNU Diffutils (https://www.gnu.org/software/diffutils/).

Auto correction performs the following procedures. First, any undesirable half-width characters, like ', ¥, <, >, |, and half-width katakana are converted into full-width characters. However, normal headings defined with H1 to H5 tags, enclosed with "<" ">" are not converted. Next, the garbled text (Mojibake) is deleted. Each character is assessed individually, as to whether it is garbled or not, and only characters that are not defined by a character encoding system are deleted. Thus, if a character is replaced with a different character that has a character code, then that character will not be deleted. Machine dependent characters and three-byte EUC-JP characters are also considered to be Mojibake, and are deleted. Auto

correction also wraps any line that contains more than 4000 full-width characters. When KH coder finds a long line, it wraps the line right after a half-width or full-width space, hyphen, dash, punctuation mark, or other such mark. If these marks are not found in an appropriate position within a line, then the line will be labelled as "cannot correct automatically," and auto correction will not be carried out.

A.4.2 [Run Pre-Processing]

Overview

Executing this command segments sentences in a target file into words, and organizes these results as a database. Given that this is a necessary preparation process for analysis, you need to execute this command when you create a new project.

However, if the text data are in Japanese, we recommend performing the "Check the target file" command, before running this process. It may also make work more efficient, if you check for any compound words that need to be intentionally extracted as single words, using the [Word clusters] command (section A.4.4), and to specify intentional word extraction in the Select words to Analyze window (section A.4.3), as required.

Pre-processing may take several minutes or more. Depending on the size of data, it may even take a few days. For this process, there are no particular settings or options you can select.

Internal processing

To segment sentences into words when analyzing Japanese text, a morphological analysis is executed using ChaSen. The output from ChaSen is stored in a form suitable for coding or searching in a MySQL database. The main tables created for data storage and relationships among these tables are shown schematically in Figure A.10. Table A.6 shows table names and column names used in each table. As described in section A.11.1, it also is possible to work directly with the MySQL database to carry out searching and tabulating.
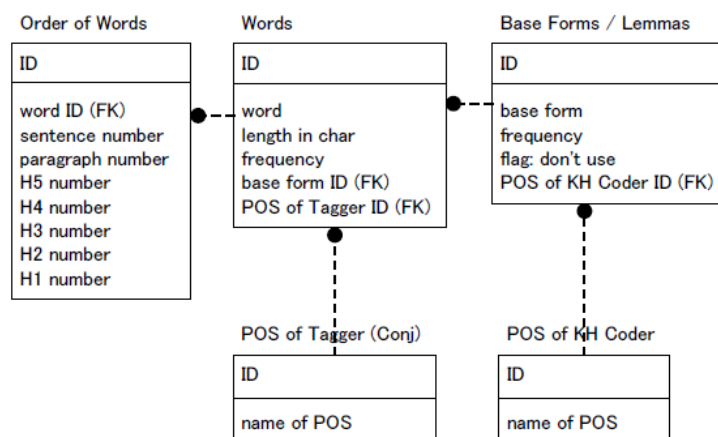

Figure A.10 Main tables and their relationships created in MySQL by KH Coder

A.4.3 [Select Words to Analyze]

Overview

Automatic word extraction in KH Coder may not always extract words, as you intend. For example, you would like to treat "personal computer" as one word, but KH Coder recognizes this as the two separate words "personal" and "computer." In another example, if a target file contains the names of

Table A.6 Names of columns in all main tables in MySQL created by KH Coder

Order of Words: "hyosobun"		
Field Info	**Field Name**	**Other**
ID	Id	Key
word ID	hyoso_id	FK
sentence number	bun_id	
paragraph number	dan_id	
H5 number	h5_id	
H4 number	h4_id	
H3 number	h3_id	
H2 number	h2_id	
H1 number	h1_id	

Base Forms / Lemmas: "genkei"		
Field Info	**Field Name**	**Other**
ID	Id	Key
base form	name	
frequency	num	
flag: don't use	nouse	
POS of KH Coder ID	khhinshi_id	FK

Words: "hyoso"		
Field Info	**Field Name**	**Other**
ID	Id	Key
word	name	
length in char	len	
frequency	num	
base form ID	genkei_id	FK
POS of Tagger ID	hinshi_id	FK

POS of KH Coder: "khhinshi"		
Field Info	**Field Name**	**Other**
ID	Id	Key
name of POS	name	

POS of Tagger: "hinshi"		
Field Info	**Field Name**	**Other**
ID	Id	Key
name of POS	name	

newspaper sections, it may be convenient to locate which articles are located in which sections. However, for statistical analysis, you may want to avoid treating such section names as words. You can use this command to customize your analyses to address such issues. Executing this command opens the window shown in Figure A.11.
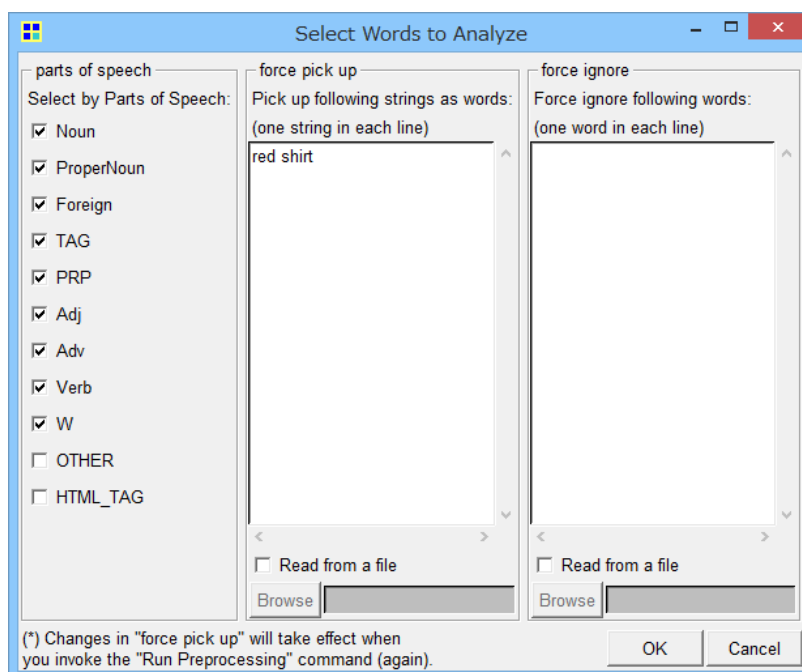

Figure A.11 Word selection window

■**Parts of speech** By unchecking any POS box on this window, words assigned to that POS are excluded from the analysis. Words that are excluded from the analysis cannot be used in searches or coding rules. However, they are counted to determine the number of words in the document.

Perhaps, it is not necessary to make any changes to this setting. Given that the POS used for analysis can also be selected under each individual analysis window. However, it is usually better to select a POS, according to the characteristics of individual analytical functions or for a specific analytical purpose. If any POS box is unchecked at this point, that POS will not be shown in the analysis window and will be excluded from the analysis. Therefore, we suggest unchecking boxes, only if you absolutely sure that a given POS is not required for any analysis.

■**Force pick up**   Phrases or strings typed in this field are forcibly extracted as one word, and given a unique POS classification called a "Tag." This function is particularly useful when an important word you want to use in your analysis is not automatically extracted as one word. To enter multiple phrases or words, write each phrase or word on a new line. The Windows version of KH coder allows one to automatically enter the contents of a file by dragging and dropping a text file into this field.

When multiple words are entered in this field, KH coder gives priority to a phrase or string higher in the list. Thus, you need to consider the order of priority, if you enter multiple phrases or strings that contain the same word or part. For example, you enter both "net usage" and "usage fee" in this field, while in the target file, there is a sentence that says, "net usage fee has decreased year after year." If "net usage" is higher in the list than "usage fee," the sentence will be divided into "net usage / fee," and it will not extract the word "usage fee." On the contrary, if "usage fee" is higher than "net usage" in the list, the sentence is segmented into "net / usage fee", and in this case, "net usage" is not extracted.

Note that any entry in this field will be lost once the project is deleted from KH Coder. Thus, it is better to keep a backup text file for such inputs. This way you will not need to directly enter phrases or strings in this field, but can specify them by checking the "read from a file" box. Now phrases or words are read from a specified text file every time pre-processing is performed. If you want to use a common setting for multiple projects, specify the same file for these projects. The file used here to enter phrases or strings for forced pick up should have the same format as used for the manual entry field.

■**Force ignore**   Phrases or strings entered in this field are not only excluded from analysis, but also considered nonexistent. They are even ignored in processes, like counting the words in a document. However, when displaying a specific document using the [Search Documents] command, phrases or words specified here are displayed as a part of the document

Suppose you want to ignore newspaper headings, such as "social section" and "economic section", then you will need to enter these words in both the "force pick up" and "force ignore" fields. These words are generally segmented into, for example, "social" and "section"; therefore, you first need to enter "social section" in the "force pick up" field, to be recognized as one word. Later, entering "social_section" in the "force ignore" field allows you to treat the heading "social section" as nonexistent in your statistical analysis.

As described above, the Windows version of KH Coder allows you to automatically enter the contents of a file by dragging and dropping a text file into this field. Note that the contents in this field will also be lost, when a project is deleted. Just like the "force pick up" field, you can make a list to read from a text file.

Internal Processing
All of the contents specified here are saved to the MySQL database. In particular, "Parts of speech" is saved to "hselection.ifuse"; "Force pick up" is saved to "dmark.name"; and "Force ignore" is saved to "genkei.nouse." Each setting is saved to the MySQL database in the form "[table name].[column name]."

A.4.4   [Word Clusters] > [use TermExtract]

Overview

The following description applies only when the target language is Japanese and ChaSen is used for word extraction. This command is also useful for finding compound words in English text data.

Morphemes recognized by KH Coder (with ChaSen) are sometimes considered too minute for word extraction. For example, "PC communication" may be extracted as two words: "PC" and "communication". To extract more than one morpheme as a single word (e.g., as in "PC communication"), you need to specify any strings in the [force pick up] field, as described in the previous section. However, before doing so, it is necessary to identify such minute or split words to define the combinations of words to be extracted as single word clusters. This command was developed for this specific purpose. Because the process is automatic, unintended word combinations may occur. For example, "PC communication" and "user", which normally would be considered as two word clusters, may be combined as "PC communication user". However, each word cluster is scored, and normally, highly scored clusters are reliable.
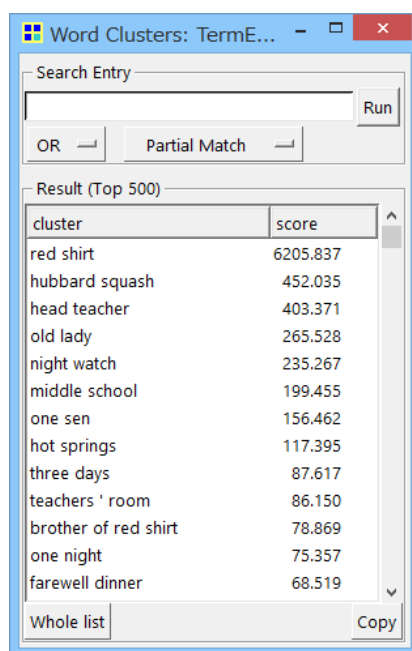

Figure A.12   Word Clusters window (using TermExtract)

Executing this command displays the window shown in Figure A.12. Clicking the [Run] button without entering any data displays the top 1,000 word clusters in descending order of scores. In addition, in the same way as [Search Words] (see section A.5.5), the window allows one to specify various search modes, such as [AND], [OR], [Partial Match], [Forward Match], [Backward Match], and [Complete Match], with the entered strings. Clicking the [Whole list] button creates and displays a list of all the word clusters obtained, including the top 1,000 data. The list is saved as a CSV file and opened with Microsoft Excel.

This command uses TermExtract, an automatic technical term (keyword) extraction system published by the Nakagawa Laboratory of the Digital Library Division, Information Technology Center of The University of Tokyo. For further details on TermExtract, refer to the website:

http://gensen.dl.itc.u-tokyo.ac.jp

TermExtract parameters are set to their default settings. This command displays only compounds of

more than one word extracted by TermExtract.

Internal processing

Executing this command, extracts technical terms (keywords) using TermExtract and then compounds are retrieved from among these extracted terms. The compounds (word clusters) and their scores are saved in the columns named hukugo_te.name and hukugo_te.num of a MySQL database, respectively. The same data sets are saved in the "coder_data" folder as a CSV file, with a name consisting of the target file name followed by "_hlte.csv". Word cluster extraction is carried out only once, on execution of this command. Thereafter, only the first extraction results will be displayed, and no new extraction will be carried out unless the timestamp of the target file is updated.

A.4.5   [Word Clusters] > [use ChaSen]

The following description applies only when the target language is Japanese. Thus, this command is for Japanese text data only.

Overview

This command was developed for the same purpose as the [use TermExtract] command under [Word Clusters] described in the previous section. Its window operation also is similar. The only difference is that this command uses ChaSen instead of TermExtract to identify word clusters. Basically, all nouns that appear in succession are combined and extracted as word clusters.

Internal processing

Executing this command changes the settings in ChaSen so that it will extract word clusters, or more specifically, compound nouns (see section A.2.3). Next, a morphological analysis is performed and a list of word clusters is created and displayed. Following these processes, the settings of ChaSen are restored. The data are saved in a MySQL table named hukugo, as well as in a CSV file in the "coder_data" folder, with a name consisting of the target file name followed by "_hl.csv".

## A.4.6   [Check the Result of Word Extraction]

Overview

This command is useful to confirm how words are extracted from sentences containing specific strings or phrases. To confirm which words are extracted frequently from the entire data, it is better to view the word frequency list (section A.5.1) or execute the [Search words] command (section A.5.5).

Executing this command displays the window shown in Figure A.13. Enter a string or phrase in the input field and click the [Search] button. The [Result] frame displays how the sentences that contain the specified string or phrase are split for the extraction of words. Select a sentence from the results and click the [Details] button or else double-click the selected sentence to see more detailed word extraction results in a Details window. The POS names of the extracted words given by the POS tagger are shown, in addition to their POS names in KH Coder.

Internal processing

This command searches for sentences that contain the entered string and then displays the results of how words are split for extraction in those sentences. Using this command, up to 1,000 sentences can be retrieved and displayed in the [Result] frame, even if more than 1,000 such sentences were obtained in the search.
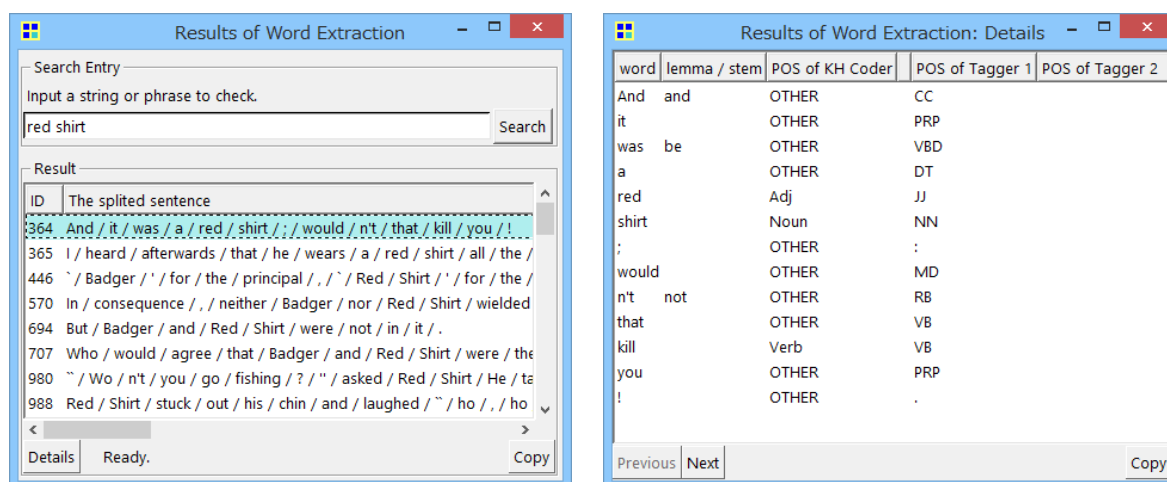
(a) Search and overview window   (b) Results window showing details

Figure A.13   Confirmation of the results of the morphological analysis

A.5   [Tools] > [Words] menu

A.5.1   [Frequency List] of words

Overview

This command enables you to view a list of words extracted by KH Coder. Selecting this menu command, displays the Options window shown in Figure A.14. Clicking the [OK] button creates and automatically opens an Excel file (*.xls) that lists the words by POS classification in descending order of their frequencies. On Windows PCs, either Microsoft Excel or LibreOffice Calc can be used to view the list presented in Figure A.15. Note that only words targeted for analysis are listed in this file.
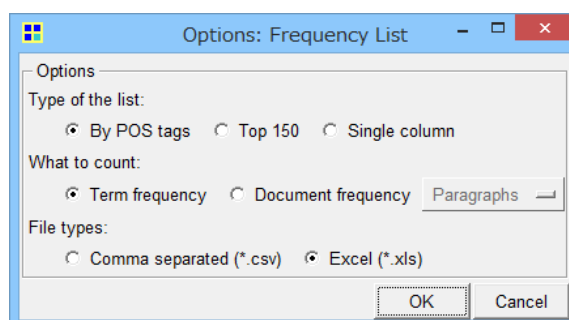

Figure A.14   Options for creating frequency lists

Options

The Options window (Figure A.14) allows a selection of different tabulation modes for the frequency list. Users can choose "Term Frequency" (TF) for the number of occurrences of each target word in the entire data; or "Document Frequency" (DF) for the number of cases containing each target word. In the latter mode, the unit to be regarded as a case, also must be specified (see section A.2.1).

CSV or Excel files can be output. Note that the 65,537th and later lines will not be output, when Excel is specified as the output format, so select wisely when the amount of data is extremely large.

"Single column" and "Top 150" can be specified as the type of list, in addition to the default setting

"By POS tags". When "Single column" is selected, words are listed in a column in descending order of their frequencies, instead of being separated into several columns by their POS tags. When "Top 150" is selected, the most frequent 150 words are listed in three columns, in descending order of abundance. This option is provided to allow easy creation of a single A4- or B5-size sheet, listing the frequently used words. Top 150 lists exclude POS consisting of many general terms, such as nouns B, adjectives B, verbs B, adverbs B, negative auxiliary verbs, and adjectives (auxiliary).

	A	B	C	D	E	F	G	H	I	J
1	Noun		ProperNoun		Foreign		PRP		Adj	
2	school	130	Red	175	tis	3	theirs	1	old	76
3	room	119	Shirt	169	utai	3			good	47
4	teacher	119	Porcupine	140	belle	2			wrong	43
5	house	108	Clown	73	banzai	1			sure	36
6	time	95	Kiyo	73	ho	1			better	33
7	fellow	88	Tokyo	47	kin	1			bad	31
8	student	85	Hubbard	46					hot	30
9	day	84	Squash	46					red	26
10	way	79	Badger	32					strange	24
11	night	70	Mr	31					little	23
12	head	65	Madonna	28					middle	21
13	face	64	Koga	24					new	19
14	man	57	Sir	21					big	18
15	principal	56	Yedo	16					second	18
16	thing	50	Hotta	15					able	16
17	town	49	Nobeoka	14					clown	16
18	matter	44	Darling	12					fellow	16
19	place	42	Master	12					clear	15
20	yen	42	Kadoya	10					hard	14
21	letter	39	Ha	9					quiet	14

Figure A.15   Frequency list for the default setting (by POS tag)

Internal processing

Executing this command creates and saves a temporary frequency list file with the name "coder_data*_temp0.csv" (*: name of the file targeted for analysis) in the same folder as the target file. If there already is such a file in this folder, then the number in the file name format (initially "0") is incremented to keep the file name unique. All temporary files will be deleted, when the same project is opened at another time. Therefore, any files to be kept must be copied and renamed with a different file name.

A.5.2   [Descriptive Stats] > [Term Frequency Distribution]

This command creates and displays a frequency distribution table of TF (Figure A.16). The command tabulates only target words for analysis. Therefore, the table is useful for identifying: (i) how many types of words will be analyzed; (ii) the mean TF of those words; and (iii) the minimum TF of words that should be included in the scope of your analysis, if you want to restrict the number of words to less than 5,000, for example.

Looking at Figure A.16, we can see that 4,204 types of words are targets for analysis, with a mean TF of 4.14. In addition, there are 2,211 types of words that occur only once (TF = 1). Furthermore, by excluding words with a TF =< 5, then the number of analysis targets can be reduced by 3,568 types (approx. 85%).

In addition, clicking the [Plot] button displays the plot shown in Figure A.17(a). Here, the horizontal axis (x-axis) indicates TFs, while the vertical axis (y-axis) shows the number of types of words corresponding to each TF. The default scale for the x-axis is logarithmic, but it can be changed to a linear scale, else both x- and y-axes can be set to logarithmic scales. The plot data is saved by clicking the [Save] button shown in Figure A.17(a). For details on how to save and use plots, see section A.5.4.
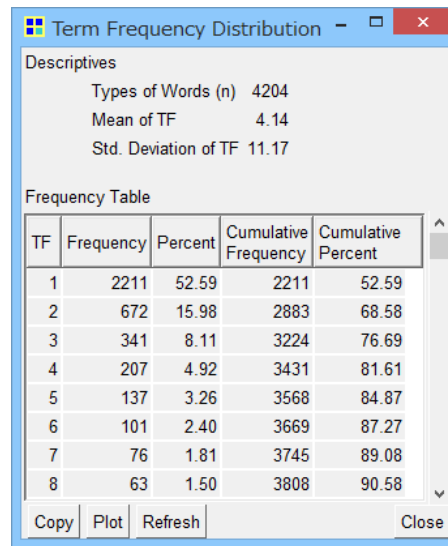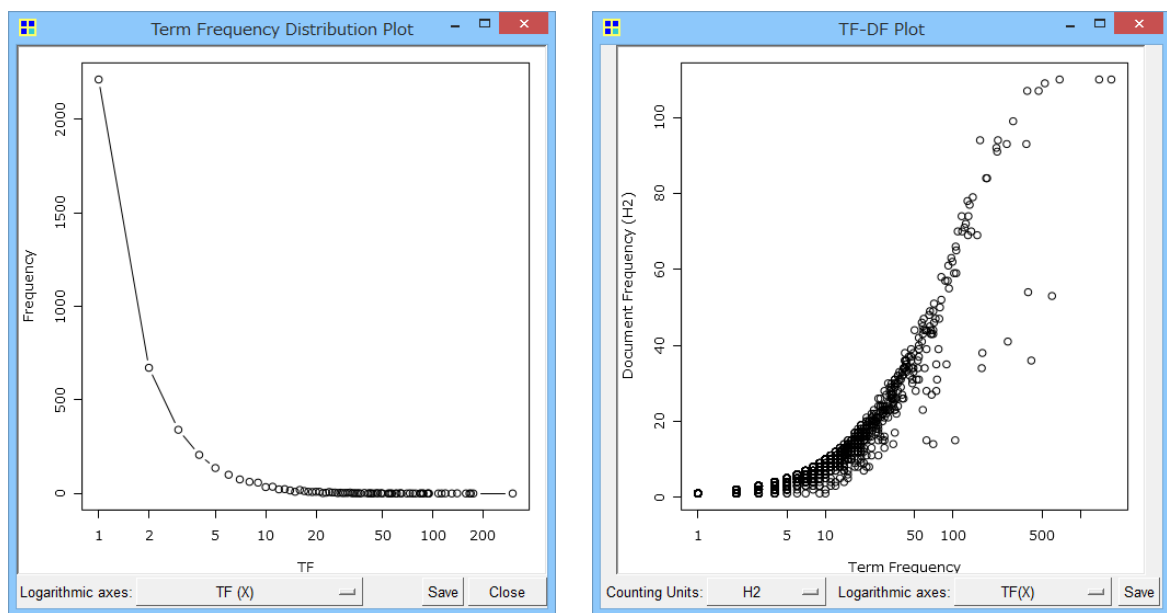
Figure A.16   Frequency distribution table for term frequency (TF)

(a) Term frequency (TF) distribution   (b) Term frequency-document frequency (TF-DF) plot

Figure A.17   Descriptive statistics of words

A.5.3   [Descriptive Stats] > [Document Frequency Distribution]

Executing this command tabulates a distribution for DF, which is the number of documents or cases that contain each term. It displays a table similar that shown in Figure A.16. Clicking the [Plot] button displays the same type of plot as shown in Figure A.17(a). This command summates the number of cases that contain each term, while the [Term Frequency Distribution] command (section A.5.2) simply tabulates the number of occurrences of each term in the text data. This command allows one to specify the unit used to identify a case in your text data (see section A.2.1).

A.5.4   [Descriptive Stats] > [TF-DF Plot]

Overview

This command allows one to evaluate the correlation between TF, the number of occurrences of each term in the entire data, and DF, the number of documents or cases in which each term is used. Normally, there is a strong correlation between these values, namely, the higher the DF, the higher the TF. However, the degree of correlation varies, depending on the data. Therefore, it is worthwhile to confirm this using the plot obtained via this command. The plot is also useful as a basis, when considering your selection of words to be used for multivariate statistics (e.g., using selections based on TF, or based on DF, or both).

Executing this command creates a plot, as shown in Figure A.17(b). The horizontal axis (x-axis) gives TFs and the vertical axis (y-axis) gives DFs. This window allows one to specify the unit used to identify a case in the text data (section A.2.1). The default scale of the x-axis is logarithmic, but it can be changed to a linear scale. Both x- and y-axes also can be set to logarithmic scales.

Saving and using plots

Clicking the [Save] button in the window shown in Figure A.17(b) saves the plot image in EMF, EPS, PDF, PNG, SVG or R Source formats. Note that the EMF format is only available with KH Coder for Windows. We suggest you specify an EMF format to paste the image to a Word or PowerPoint file, but use an EPS format to paste the image to a LaTeX file. Both EMF and EPS are vector graphic formats, allowing smooth printing of characters, curves, and oblique lines. The PNG format is appropriate for web pages, or plots viewed on a computer screen.

KH Coder uses the R software environment for statistical computing and graphics to create plots. If the R Source format is selected, when the plot is saved, then the plot creation command that is sent to R by KH Coder will be saved. Thus, it is possible to view the R commands by opening this saved file, allowing one to verify the statistical processing and plotting. In addition, executing these saved commands within the R environment generates the same plot, created in KH Coder. Moreover, all plots can be customized by modifying their command options in R. This allows flexible settings for statistical options and plot drawing options that are not available in KH Coder.

Figure A.18 presents an example of such customization using the R environment. This plot is the customized version of the saved TF-DF plot (Figure A.17(b)) file. After the dot color and shape were changed, the "identify" command was used to detect words with a relatively high TF, in spite of their low DF (i.e., the number of chapters containing these words was relatively small). The text data used for this example was the novel Kokoro by Soseki Natsume, also shown in the Quick Start Tutorial. Almost all names of the main characters featured in this novel are to be found in Figure A.18.

A.5.5   [Search Words]

Overview

Executing this command opens the window shown in Figure A.19, allowing the user to search words extracted by KH Coder. The search results are displayed in descending order of frequency. If the words have conjugated or declensional forms, their forms and corresponding frequencies also are displayed. Here, frequency refers to the number of occurrences of the term in the entire text. Clicking the [KWIC Concordance] button with the target word selected in the Result frame, or else simply double-clicking the word, displays its KWIC Concordance window (see next section).

Options
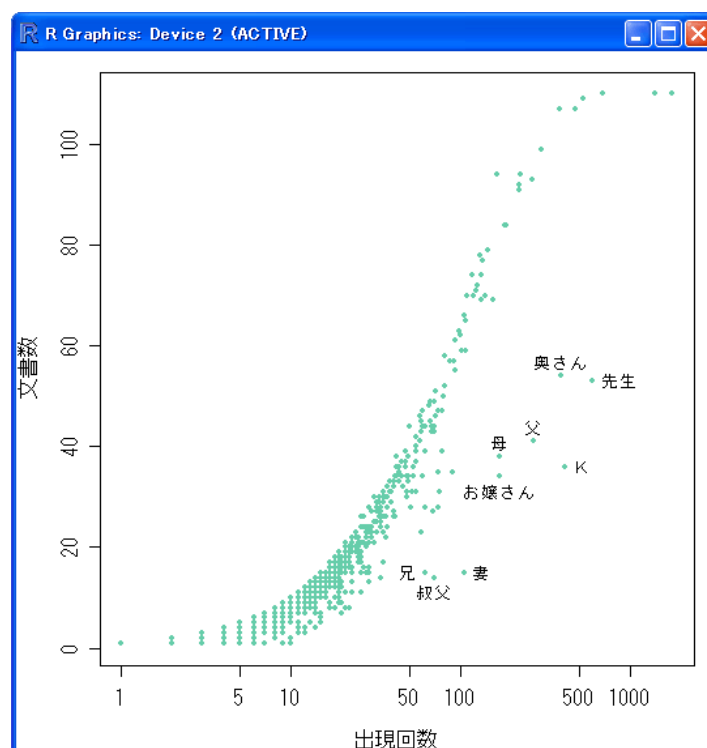
Figure A.18   Customization of a plot in the R environment
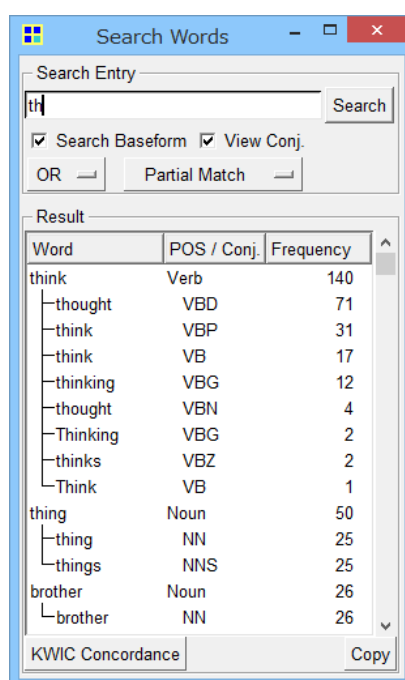

Figure A.19   Search Words window

■**Search Entry**   When the [Search] button is clicked or the Enter key is pressed after a string has been entered, a search is performed. Spaces can be used to delimit plural strings, using either single- or double-byte spaces.

■**Search Baseform**   When this option is selected, the words extracted by KH Coder are searched. When this option is deselected, the search targets are not the words extracted by KH Coder, but the morphological analysis results obtained with ChaSen. Naturally, in the latter case, the POS names and conjugation/declension forms output by ChaSen are displayed, instead of the POS names given by KH Coder.

■**View Conj.**   Deselecting this option disables the display of conjugations/declensions of each word, and their corresponding frequencies.

■**AND/OR search**   This option allows the user to specify an AND or OR search phrase, when two or more strings are entered in the Search Entry field, delimited by a space. Clicking the inset button beside "OR", shown in Figure A.19, toggles between AND and OR. When AND is selected, the words included in all the strings entered are searched, while when OR is selected, only those included in at least one string are searched.

■**Search methods**   Clicking the inset button beside "Partial Match", shown in Figure A.19, allows the user to select the desired search methods from four options:

Partial Match: All the words containing the entered strings are searched.
Complete Match: Only the words identical to the entered strings are searched.
Forward Match: The words starting with the entered strings are searched.
Backward Match: The words ending with the entered strings are searched.

A.5.6   [KWIC Concordance]

Overview

This function enables you to analyze how an extracted word is used in the target file. This is illustrated for the word "say" in Figure A.20. If desired, then the search can be narrowed to only display a specific POS classification or a specific conjugation of the word. It is also possible to search for only a POS or a conjugation, and not a word. However, if the amount of data is very large, then searching for "Noun" can generate tens of thousands of hits, which will take a long time to process. The search function and results indicated in the window shown in Figure A.20 are generally referred to as "KWIC: Key Words in Context," or simply "concordance."

If there are more than 200 search results, then only the first 200 are initially displayed. Additional results can be viewed by clicking the [N200] or [P200] buttons to browse the next 200 or previous 200 results, respectively. A return character is replaced by the ' ↓ ' symbol on the screen. However, return characters immediately after lines enclosed in HTML tags H1 to H5, i.e., heading lines (section A.2.1), are not displayed. Only return characters that delimit paragraphs are shown.

Options

■**Displaying a broader context**   To see a broader context to a word than that displayed in the KWIC Concordance window (Figure A.20), double-click a search result or select a search result and click the [View Doc] button to open the Document window (Figure A.36(a) in A.6.1). In the KWIC Concordance window (Figure A.20), it is also possible to select the document range ([Sentences], [Paragraphs], or [H1] to [H5]) to be displayed by clicking the button to the right of [Units].

■**Sort order**   The search results are sorted as follows: first, according to the condition specified in [Sort 1] condition, then items in the same set are resorted according to the [Sort 2] condition, finally, the
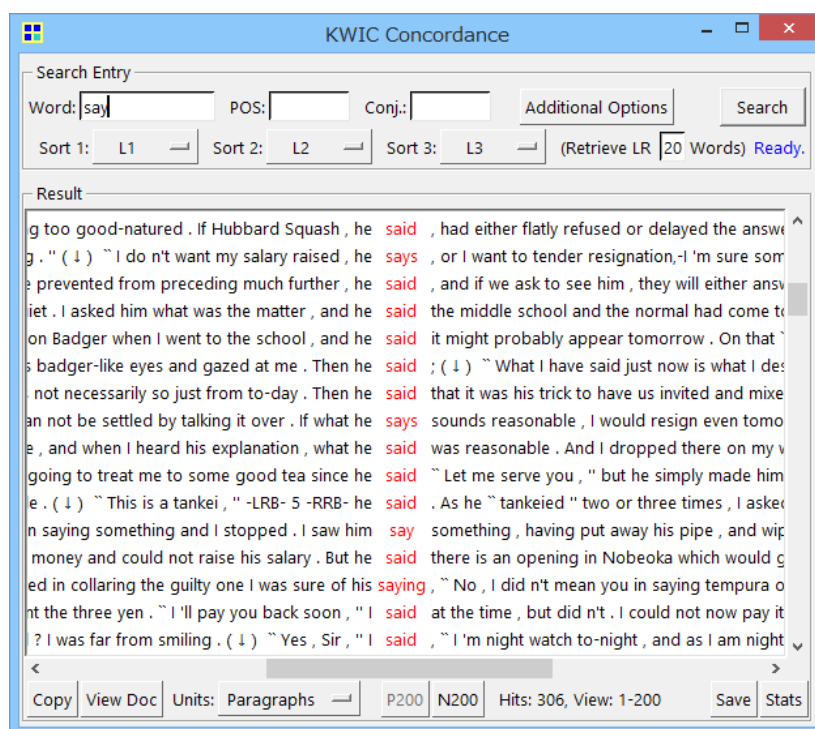
Figure A.20   Window showing KWIC concordance results

items still in the same set are resorted according to the [Sort 3:] condition. Items in the same set for the condition specified in [Sort 3:] are displayed in their order of appearance in the text.

The conditions that can be specified include "L1" to "L5" and "R1" to "R5" as well as "None", or "Conj.". If "None" is selected, then the search results are shown in the order they appear in the target file. In this case, because there are no items in the same set, meeting a given condition, then no further conditions can be specified to refine the search order. If "Conj." is selected, then the search results are listed in order of the conjugations of the search words. Choosing "L1", sorts search results with respect to the word to the immediate left of the search word. Similarly, "L2"sorts the search results with respect to the second word to the left of the search word. "L3" to "L5" function similarly. "R1" to "R5" sort the search results in the order of the word to the immediate right, through to the word located fifth right of the search word. Results are presented in descending order of frequency of appearance, rather than in syllabary order.

Please note that changing the sorting order does not initiate any new sorting. Hence, after setting the sort parameter conditions, click the [Search] button to apply them.

■**Additional options**   It is possible to narrow the concordance search results by adding options that indicate when certain words appear immediately before or after the search word. Clicking the [Additional Options] button at the top of the KWIC Concordance window (Figure A.20) opens the new pop-up window shown in Figure A.21. Specify the desired options in this window and click [OK]. Clicking [Search] in the KWIC Concordance window again (Figure A.20) will perform the search with the additional options. If additional options were specified, then the [Additional Options] button in the KWIC Concordance window changes to [Additional Options*] (with '*' appended at the end). Note that you must cancel any additional options used for previous searches, before beginning a brand new search; not doing so will lead to undesired results.

When specifying additional options (Figure A.21), up to three conditions can be added to your search
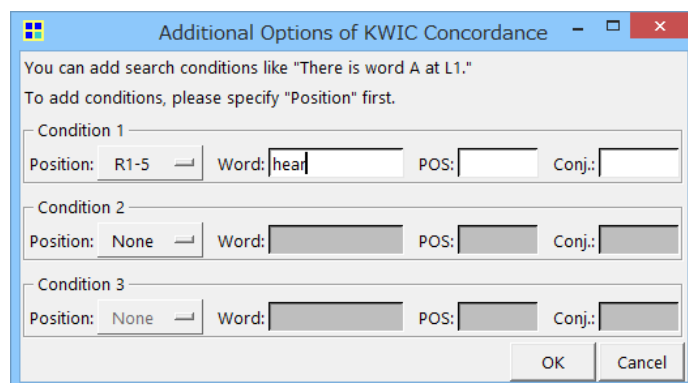
Figure A.21   Additional concordance options

parameters. To add a condition, first specify the "Position" in the first column. The specification format used here is similar to that used for the sort order in the KWIC Concordance window (subsection above), with options ranging from "L1-5", "R1-5", and "None". A number of examples are described below. "L1" indicates the word to the immediate left of the search word, i.e., the word just before it; "L3", the word third to the left; "R1", the word to the immediate right, i.e., the word just after it; and "R5", the word fifth to the right. "RL1-5" specifies a word within the first to fifth words before or after the search word, and "L1-5, a word within the first to fifth words before it.

The entry for "Word" can be specified in the second column. To further narrow the search results related to this word condition to only to a certain POS or a specific conjugation, fill in "POS" and "Conj." for each specified word. For example, to search for the word "hear" appearing within the first five words after the search word, specify "R1-5" as the "Position" and "hear" as the "Word". Alternatively, to narrow a search to "saying" (present participle or VBG), specify "VBG" in "Conj." If desired, then the search results can also be narrowed, without specifying "Word," but only "POS" and "Conj.". For example, it is even possible to add a condition like "there should be a base form of the verb (VB) just after the search word (R1)."

■**Saving the search results**   Clicking the [Save] button on the KWIC Concordance window (Figure A.20) saves the results of the search currently displayed on the screen. Even if the number of results exceeds 200 and only a part of the results appears on the screen, all results are saved to a CSV file, as shown in the example in Figure A.22. Search results saved in CSV format can be displayed using a spreadsheet software like Microsoft Excel.
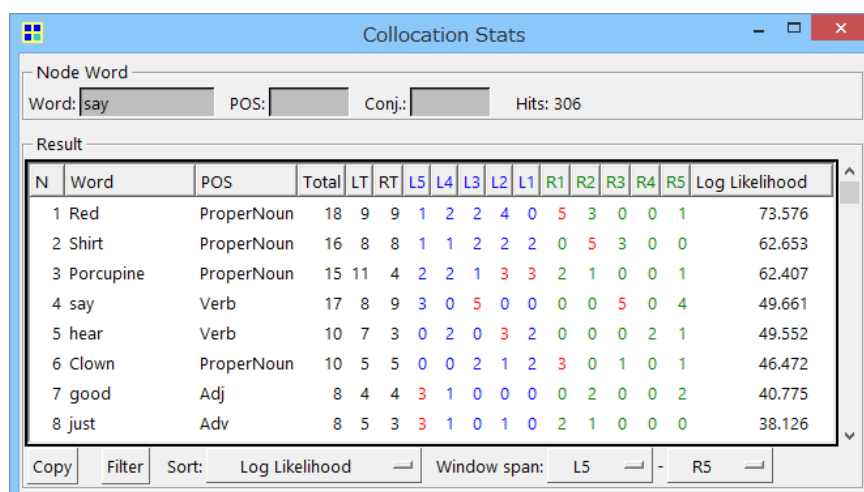

Figure A.22   Example of a saved KWIC Concordance search result

For each result, columns are created for various variables that indicate its position. Columns "h1" to "bun" provide the same information, as those generated using the command [Export Document-Word Matrix] (in menu Tools > Documents; see section A.6.7). Both "bun-No." and "mp-No." are numbers

used to indicate the positions of the node-word (e.g., word "say" in Figure A.20). The variable "bun-No."
gives the sequence number of each sentence associated with the search word, counted from the beginning
of the target file. In "bun-No.", all headings enclosed in H1 to H5 tags are counted. The "mp-No." is
the sequence number for all words counted from the beginning of the target file, even if a word is not
the target of analysis.

■**Collocation statistics** If a concordance search generates millions of lines of results, it is not easy
to check them line by line. To help the user obtain an overview of results, collocation statistics can
be generated and displayed by clicking the [Stats] button on the KWIC Concordance window (Figure
A.20). Using the Collocation Stats window (Figure A.23), it is easy to determine which words appear
frequently before and after the target word (or node word). In Figure A.23, the statistics show that the
word "here" appears thrice in a position two words before (L2) and twice just before (L1) the node word
"say". In addition, we see clearly that words like "Red", "Shirt", and "Porcupine" are often used in
association with "say". Using these results, more refined searching is possible by specifying [Additional
Options]. For example, these options can be used to generate results, where the word "hear" appears
two words before (L2) the target word.

Figure A.23 Collocation statistics window

Words listed in the Collocation Stats window are often related to the node word. Therefore, this
window can be used to identify words having a strong relationship with the search word. Since these
results list the words that frequently appear within five words of the node word, the list tends to show
the words that have direct dependency on the node word. In contrast, executing the [Word Association]
function (see section A.5.7) identifies words that appear in the same document with high probability,
regardless of whether they have direct dependency on the search word. Both search functions are useful
for different purposes.

Given that the Collocation Stats window (Figure A.23) is linked to a specific [KWIC Concordance]
search result, performing a new concordance search updates the Collocation Stats window, accordingly.
The order of the words displayed in the Collocation Stats window can be changed by clicking the button
to the right of the label "Sort:". For example, selecting "L1" sorts and arranges the words in descending
order, using the frequency with which the words appear one word before the node word. The default
score is calculated using the function $f(w)$ shown below, where l1 is the frequency of a certain word w
that appears just before the node word; l2 is its frequency, two words before the node word; r1 is its
frequency just after the node word; and r2 is its frequency, two words after the node word.

$$f(w) = \sum_{i=1}^{5} \frac{(l_i + r_i)}{i}$$

In general, the greater the frequency that a certain word w appears before or after the node word ($li + ri$), the larger the value $f(w)$. In calculating the value $f(w)$, frequencies ($li + ri$) are divided by "i," which weighs the frequencies according to their distance from the node word. Thus, words that appear nearer to the node word (i.e., with a smaller "i") have greater weight than those that occur five words before or after the node word. In this formula, the frequencies of words that appear just before and after are simply added, since they are divided by unity.

In addition to calculating the score based on $f(w)$, this command allows the user to obtain results using various scales, including Mutual Information, MI3, T Score, Z Score, Jaccard, Dice, and Log Likelihood. Each scale can be used to indicate the collocation between word w and the node word. In principle, calculations with those scales are carried out after the method employed in WordSmith Tools (Scott 2001), using the formulas outlined below.

$$Mutual\ Information = \log_2 \frac{aN}{F_1 F_2}$$

$$MI3 = \log_2 \frac{a^3 N}{F_1 F_2}$$

$$T\ Score = (a - \frac{F_1 F_2}{N}) \div \sqrt{a}$$

$$Z\ Score = (a - \frac{F_2}{N - F_1} F_1 S) \div \sqrt{(\frac{F_2}{N - F_1} F_1 S)(1 - \frac{F_2}{N - F_1})}$$

$$Jaccard = \frac{a}{F_1 + F_2 - a}$$

$$Dice = \frac{2a}{F_1 + F_2}$$

$$\begin{aligned} Log\ Likelihood = {} & a \ln a + (F_1 - a) \ln (F_1 - a) + (F_2 - a) \ln (F_2 - a) \\ & + (N - F_1 - F_2 + a) \ln (N - F_1 - F_2 + a) \\ & - F_1 \ln F_1 - F_2 \ln F_2 \\ & - (N - F_1) \ln (N - F_1) - (N - F_2) \ln (N - F_2) \\ & + N \ln N \end{aligned}$$

where a is the frequency of word w, before and after the node word within the window span; $F1$ is the frequency of the node word in the entire target data; $F2$ is the frequency of word w in the entire target data; S is the width of the window span (in the case of the default this ranges from [L5] - [R5], so the value is 10); and N is the token count (total number of all words contained in the entire target data).

It is also possible to select words to be listed using their total values, e.g., by specifying a minimum frequency within five words before and after the node word.

Clicking the [Filter] button opens the window shown in Figure A.25, used for choosing which words to display. It is possible to list words with a given POS classification, or to list only words that appear frequently throughout the entire document.

A.5.7   [Word Association]

Overview

This command enables you to find words that are closely associated with a specific word, as well as words that are closely associated with a specific code. Since coding rules enable you to specify many

types of conditions, this command can be used in many different ways. For example, in a text document compiled of responses to an open-ended survey question, creating a code that represents the condition "the respondent must be male" picks up words that are closely related with the variable "male", i.e., typically appear in male responses.

These associations are estimated from data using conditional probability calculations. Thus, when specifying conditions such as "a specific word must appear" and "a specific code must be attached," the documents that satisfy these conditions are retrieved internally, then the words that frequently appear in those documents are listed in the window. In Figure A.24, the words closely related to the word "teacher", or rather the words most likely to appear in documents that contain "teacher" are: "head," "student," "history," etc.
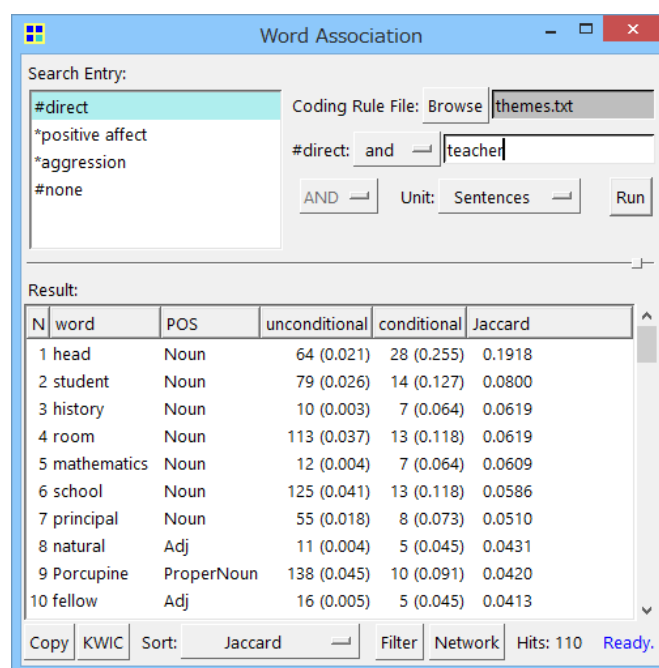

Figure A.24   Word Association window showing results for "teacher"

As Figure A.24 shows, this command not only lists closely associated words, but also yields much additional information. "Hits" shown on the bottom right shows the number of documents satisfying the specified conditions. The "POS" column lists the POS classification for each word; the "unconditional" column indicates the number of documents in which each word appears in the target file, and the probability of its appearance throughout all documents (unconditional probability); while the "conditional" column lists the number of documents satisfying the specified conditions, in which each word appears, and the probability of appearance in documents satisfying the specified conditions (conditional probability). Finally, the rightmost column determines the selected display order and labeled as "Jaccard", shows the values used for sorting the results (display order). The values in this column will be discussed later in this section.

Options

■**Search Conditions**   The procedure for specifying the search conditions, such as "a specific word must appear" and "a specific code must be attached" is the same as that used in the [Search Documents] window (see section A.6.1). The options in the upper right corner enable you to perform a search using codes read from a coding rule file. You can also simply search by entering a word directly, and selecting

the "#direct" code on the left hand side. It is also possible to specify the "Unit"— sentences, paragraphs, or H1-level headers—to define the range within a single document for searching and tabulating.

■**Display order: [Sort]** To list words that appear frequently in documents that have a specific word or code, you can currently select from five different types of display orders: "Differential", "Lift", "Jaccard", "Ochiai", and "Frequency". To change the display order, click the menu button to the right of "Sort:" in Figure A.24.

The option "Differential" uses the difference between the unconditional and conditional probabilities to order results, whereas "Lift" is based on the conditional probability divided by the unconditional probability. Both values indicate how much higher the conditional probability is than the unconditional probability of a given word for the given search criteria. The "Differential" option tends to list a relatively large number of words in common use, or words with a larger DF, i.e., words that appear in many documents. The "Lift" option, however, tends to list uncommon words that appear only in a limited number of documents. "Jaccard" and "Ochiai" are the names of similarity measures used for measuring the association between two variables. "Frequency" here simply uses the conditional probability.

None of these options list words with conditional probabilities lower than their unconditional probabilities. This is because retrieving words that appear only with probabilities equal to or lower than the probability of their appearance in the entire text is not the purpose of this command. This command is best used to search for closely associated or distinctive words.

■**Filtering Results: [Filter]** Clicking the [Filter] button opens the window shown in Figure A.25. You can use this window to filter results, e.g., to display only adjectives, or to list words that appear in a certain or large number of documents.
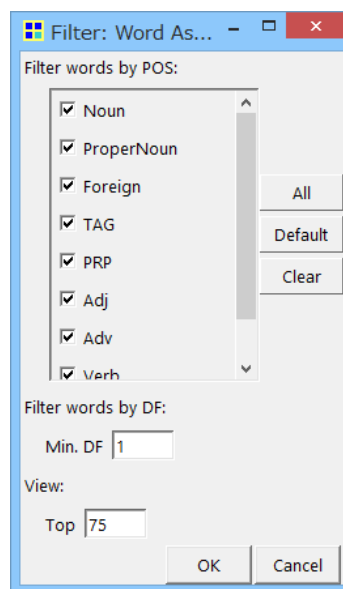
Figure A.25 Filtering for Word Association results

When you analyze Japanese text data using the default setting, "名詞 B" (Noun B), "動詞 B" (Verb B), "形容詞 B" (Adjective B), "副詞 B" (Adverb B), "否定助動詞" (Negative Auxiliary Verb), and "形容詞（非自立）" (Adjective (auxiliary)) are automatically removed from the results.

■**Co-occurrence network: [Network]** Clicking the [Network] button displays a co-occurrence network chart (Figure A.26) based on the retrieved words. In this chart, words closely associated with each other are connected with lines, while words that define the search condition are enclosed in double

rectangles. As Figure A.26 shows, you can use this command to draw a network of words closely related to a specific word. Using other variables, you can use this command to plot, for example, a network of words distinctive to female respondents in a questionnaire.

The degrees of association between words are calculated solely based on the documents that satisfy the search conditions. In the example used to generate Figure A.26, the word association is determined from only 110 documents that contain the word "teacher" (Figure A.24). To draw a network of words distinctive to female respondents, association is evaluated only for the female responses to the question-naire. The co-occurrence network generated from these data displays not only words that are distinctive to the female respondents, but also any close associations found between words in female responses have lines connecting them.

The procedure for creating the chart is the same as for the [Co-Occurrence Network] command (see section A.5.11). Clicking the [Config] button on the window in Figure A.26 and specifies similar options to the [Co-Occurrence Network] command.
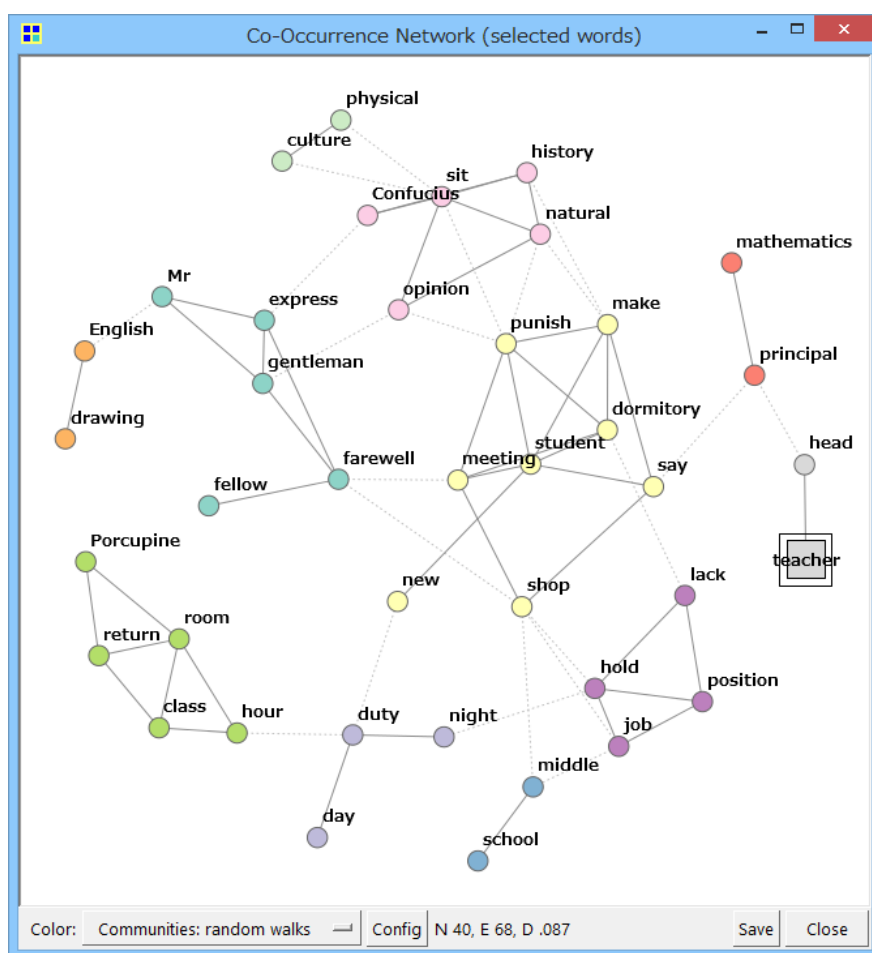

Figure A.26   Co-occurrence network of words closely associated with "teacher"

A.5.8   [Correspondence Analysis] of words

Overview

This command performs a correspondence analysis on extracted words and produces a two-dimensional scatter diagram to help visualize the results. This command is used to explore what kinds of words

have a similar appearance pattern. Variables also can be used in this command (see section A.8.1). For example, in data on responses to an open-ended survey question, you can analyze differences in word usage among age groups, and determine which age groups have similar response content. To carry out this analysis, you have to load a variable that describes each respondent's age group in advance, so you can select it when you execute this command.
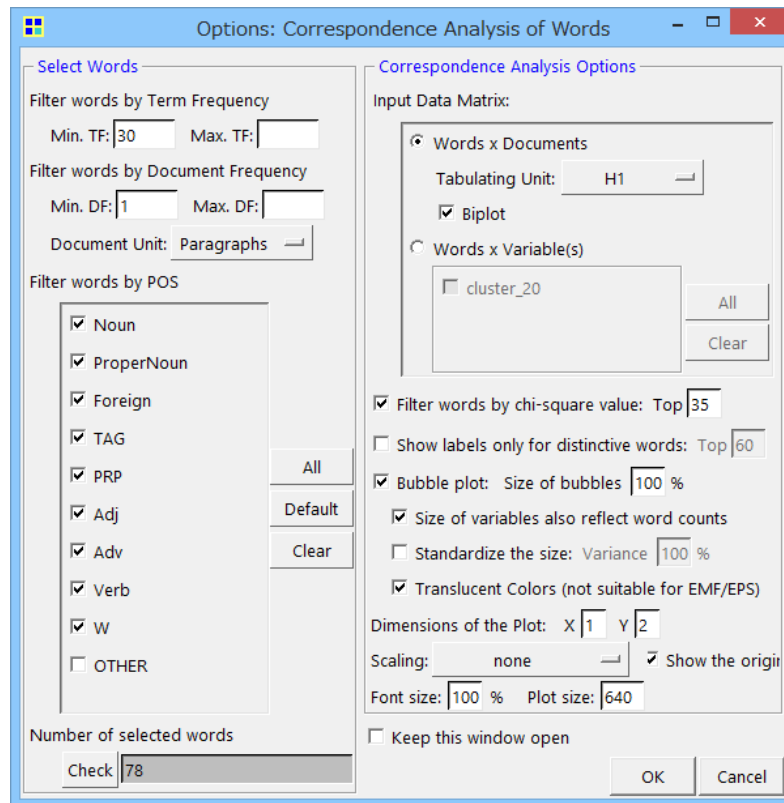
Figure A.27   Correspondence analysis options

Options

■Word selection   It is important to carefully consider word selection for this command, since plotting too many words at once on a two-dimensional scatter diagram can result in a screen filled with black, which can make it difficult to read the characters. To resolve this problem, the number of words should firstly be reduced by filtering. In the Options window (Figure A.27), you can filter words using three criteria: "TF", "DF", and "POS" classification. Clicking the [Check] button shows the number of words selected after filtering.

Applying TF and DF options can filter out words that seldom appear, as well as words that appear too frequently. Words that rarely appear are not suitable for statistical analysis. Words that appear too frequently are often too general, making them difficult to analyze. In many cases, using these options to screen out these words facilitates analysis.

In the default window, a number multiple of 5 (e.g., 30) has been automatically entered at "Min. TF". This sets the number of words to be analyzed to around 75. If you are analyzing Japanese text data, then checkboxes for "名詞 B" (Noun B), "動詞 B"(Verb B), "形容詞 B" (Adjective B), "副詞 B" (Adverb B), "否定助動詞"(Negative Auxiliary Verb), and "形容詞（非自立）" (Adjective (auxiliary)) are unchecked. Depending on the purpose of the analysis, the default screen values can be modified, as desired.
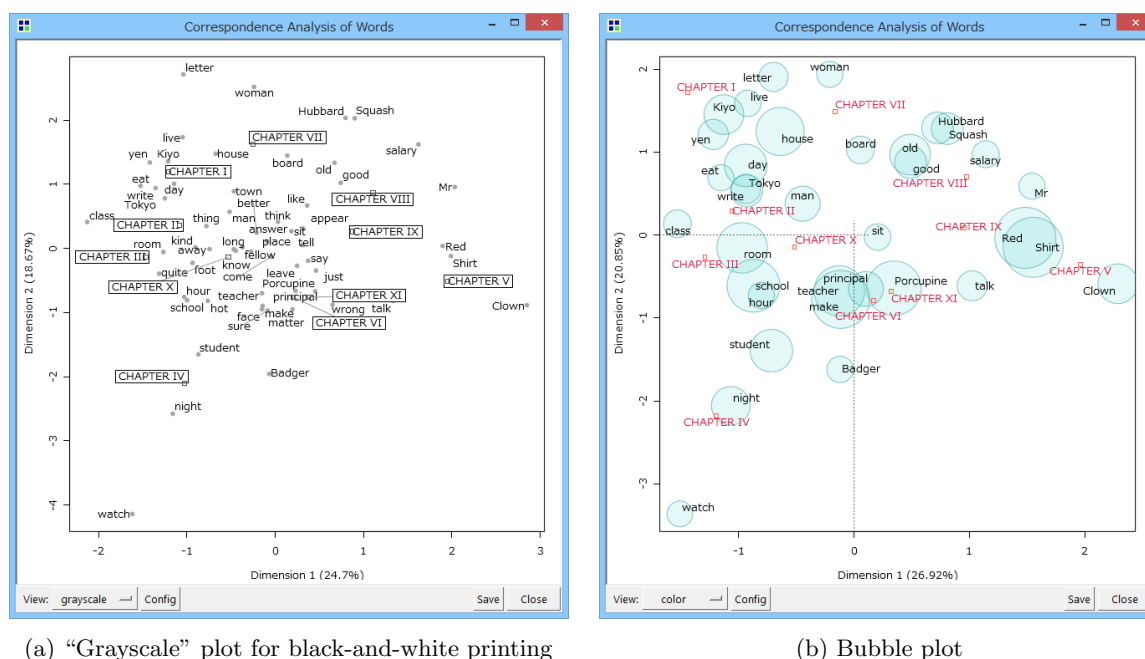
(a) "Grayscale" plot for black-and-white printing   (b) Bubble plot

Figure A.28   Results of a correspondence analysis

■**Input data matrix**   Two types of input data matrices can be selected as the starting point for your correspondence analysis: "Words × Documents" and "Words × Variable(s)". The "Words × Documents" type is best for exploring words with a similar appearance pattern, and to determine what dimensions can be found from the appearance pattern of each word. However, to see which words frequently appear under each heading—using H1 to H5 tags (section A.2.1) in the target file—we recommend selecting this option on the right under tabulating unit, and checking the box "Biplot". This matrix type works best when you select a unit that divides the entire data into less than twenty components. This is because plotting 100, 200, or more headings (or document numbers) makes it impossible to read the chart.

When you select this type of matrix, a matrix similar to that generated by the [Export Document-Word Matrix] command (section A.6.7) will be analyzed. The only difference between these two matrices is that the [Export Document-Word Matrix] command omits variables representing positions and document lengths.

To use the "Words × Variable(s)" option, designate the variables to be used in the panel. For each value of a given variable, this function summarizes the number of appearances of each word in the document that have this value. Similar to the "Words × Documents" matrix above, this matrix lists the extracted words in columns, while the values of the selected variable are along rows.

For example, if multiple variables, such as "gender" and "academic background" are selected, then the values of those variables, such as "male", "female", "university", "high school", "junior high" are listed along rows (in the first column of the matrix). Such analyses using a data matrix are called multiple correspondence analyses. They enable you to explore the association between extracted words and multiple variables, simultaneously.

■**Tabulating unit**   In a correspondence analysis, the selection of the tabulating unit (see section A.2.1) is important in the "Words × Documents" option. This is because when assessing the similarity of occurrence between words, whether these words appear frequently in the same "document" serves as the basis for analysis. Thus, the results of analysis depend heavily on the range of a single document or tabulating unit. In this step, caution should be used when deciding whether a "document" is a paragraph

or larger unit, like a chapter or article. If "Paragraphs" is selected, then the tabulation is performed in paragraphs, and groups of words frequently appearing in the same paragraph are evaluated as having a similar appearance pattern. Similarly, when tabulating in units of articles, words that frequently appear in the same article have a similar appearance pattern.

■**Configurable options**   An option also exists to draw a bubble plot, as shown in Figure A.28(b). This chart highlights words that occur more often with larger circles. If the bubble (circle) sizes are not standardized in the bubble plot, then the area of each bubble is proportional to word frequency. The bubble size for a given value of a variable or heading is determined based on the total number of words contained in the document, which all have the same value or heading. When creating a bubble plot, it is helpful to fill the bubbles with a translucent color, as shown in Figure A.28(b). Coloring the bubbles makes it easy to distinguish their size. Moreover, using a translucent color increases the visibility of their overlaps. However, if plots are saved as EPS or EMF formats, then an unfilled plot is stored because translucent colors cannot be used in those formats.

In many correspondence analyses, the area around the origin is often packed with indistinct words. In this case, the "Filter words by chi-square value" option can be used to screen out these words from analysis. This option selects and analyzes words, whose frequencies greatly change in the input data. The degrees of change in frequency are measured using the value of χ^2. The option "Show labels only for distinctive words" can also be selected to improve the legibility of the scatter diagram in this region, by omitting labels from the display, rather than screening out the words themselves.

Other options also are available for customizing the extracted dimensions used for the x- and y-axes, the font size, and plot size. As shown in Figure A.28(b), it is possible to select whether or not to indicate the origin with dotted lines. In addition, the standardization method for scaling can be selected from among three options: [none] (standardized), [symmetric] and [symmetric biplot]. Conventionally, KH Coder has used [none] as the default. This method uses the standardized scores output by the "corresp" function in R, such that plotting is performed, without fixing the aspect ratio to 1:1. Thus, results are displayed to the limit of the plot area, i.e., enlarged both vertically and horizontally. In effect, this is advantageous to quantitative text analysis that needs to display as many word labels as possible. To create plots called symmetric maps, which are used more traditionally, select the [symmetric] option. This method is the default setting for the statistical discovery software JMP developed by the SAS Institute (NC, USA). Finally, [symmetric biplot] is the default method used in SPSS (IBM, NY, USA). For details on these three methods, the user is referred to M. Greenacre (2007).

The above options can be changed by clicking the [Config] button on the results window (Figure A.28), once your analysis has been completed.

■**Views**   The results window (Figure A.28) includes a "View" option on the bottom left. Clicking the menu button just to the right of "View" and selecting "grayscale" generates a plot suitable for black-and-white printing, as shown in Figure A.28(a). Here, the headings and values are enclosed in rectangles for emphasis, instead of being colored (as in the default). Selecting "variables" will display only the labels of variable values or document numbers, but not the labels of extracted words. When analyzing more than several hundred words, displaying word labels will result in a screen filled with black. Hence, for large numbers of words, you can select "variables" as the "View" type to avoid this problem.

■**Saving the results**   The [Save] button on the bottom right of the window (Figure A.28) saves the two-dimensional scatter diagram in one of five formats: EMF, EPS, PDF, SVG, PNG, or R Source. For more information on selecting formats and using saved diagrams, see the subsection under the [TF-DF Plot] command in section A.5.4.
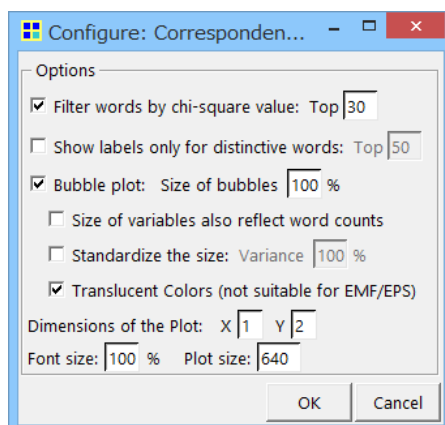
Figure A.29   Configure: Correspondence Analysis of Words window

A.5.9   [Multi-Dimensional Scaling] of words

Overview

This command enables you to carry out multi-dimensional scaling on the extracted words and to draw the results in 1- to 3-dimensional scatter diagrams. You can use this function for finding combinations or groups of words that have similar appearance patterns.

This analysis uses the table generated with the [Export Document-Word Matrix] command (see section A.6.7), with the variables that indicate positions and document lengths removed. However, if any pair of words, with a distance of zero occurs in the analysis, then one of these pairs is automatically screened out of analysis, resulting in a message indicating this step. The processes involved in this analysis are displayed in text form, from start to end on the console screen.

Options

■**Method**   The menu lets you choose from three analysis methods: [Classical] (Metric Multi-Dimensional Scaling), [Kruskal] (Non-metric Multi-Dimensional Scaling), and [Sammon] (Sammon's Non-linear Map). The default method [Kruskal] is the most widely used. [Sammon] tries to keep the distances between words from getting crowded and is recommended for cases, when [Kruskal] creates a map too difficult to read.

■**Distance**   For analyzing sparse data, where each document contains a small number of words and each word appears in only a limited number of documents, we recommend using the Jaccard coefficient to determine the associations between words. The Jaccard coefficient emphasizes whether or not specific words co-occur. This is valid, regardless of whether the word appears once or ten times in a document. Thus, word co-occurrence is calculated irrespective of appearance frequency. Even when many documents have neither word A nor word B, the Jaccard coefficient between A and B does not increase; thus, the fact that some documents contain neither word A nor word B is not considered to indicate a close association between word A and word B. For this reason, the [Jaccard] default option is suitable for analyzing sparse data (Romesburg 1984=1992).

In contrast, if documents are long and contain many words, and their frequencies in each document are important, then other "Distance" options are more suitable, e.g., [Euclid] (Euclidean distance) and [Cosine] (Cosine coefficient). If either of these options is selected, then the calculation uses the number of occurrences per 1,000 words (adjusted frequency), rather than the number of occurrences in the documents themselves (raw frequency). This prevents the document length from affecting the calculation.
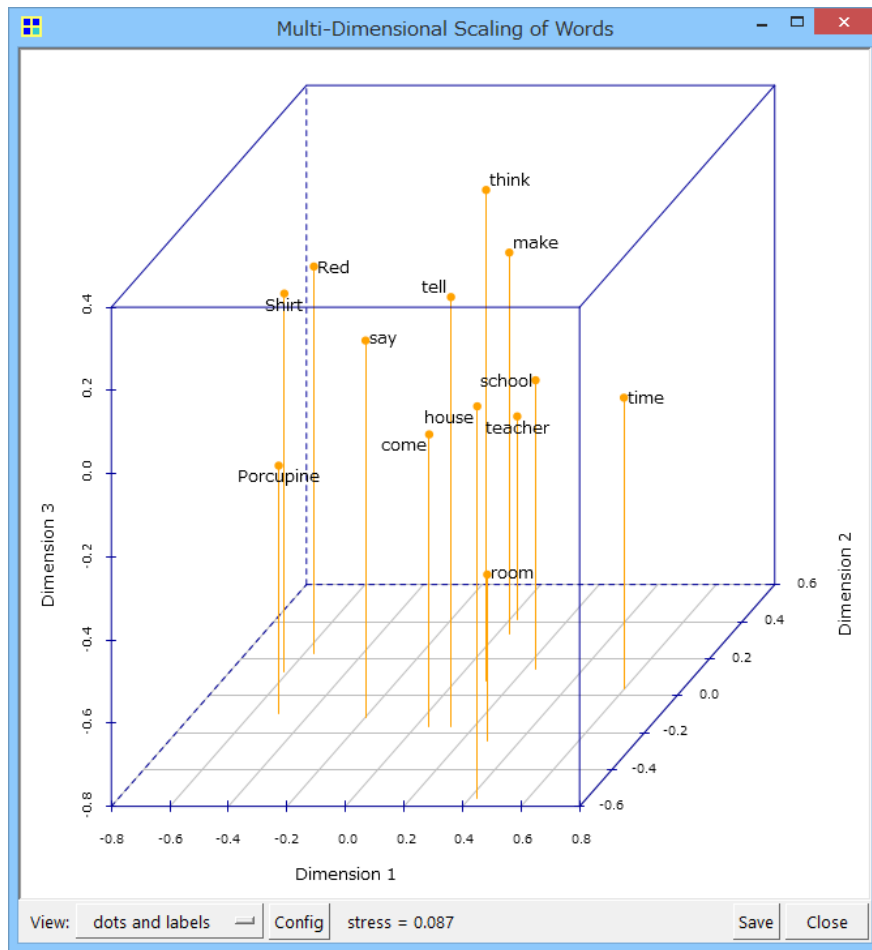
Figure A.30   Example of a 3-dimensional solution created by the Multi-Dimensional Scaling command

Moreover, if [Euclid] is chosen, the word frequency (adjusted frequency) is standardized prior to the distance calculation. This ensures the distance of the words is calculated based on occurrence patterns, rather than on whether each word appears frequently or not.

■**Coloring clusters**   Performing multi-dimensional scaling with many (50 to 100) words results in a plot that may be difficult to interpret. To make interpretation easier, KH Coder can perform a cluster analysis of words and present these clusters using different colors. Given that only 12 colors can be used at a time to identify clusters on the screen, a maximum of 12 clusters can be specified. This function is not available for three-dimensional plots.

If [Adjacent clusters] is checked under the "Multi-Dimensional Scaling Options" area on the right, cluster analysis is performed based on the score obtained in multi-dimensional scaling. This cluster analysis uses the Ward method based on Euclidian distance. Thus, words plotting close to each other are classified in the same cluster. Clearly, clustering can assist interpretation of the results of multi-dimensional scaling. This type of cluster analysis is also applied in "Concept Mapping" proposed by W. M. K. Trochim (1989), and is thought to be an effective method for exploring data with an unknown group or category structure (Afifi & Clark 1996).

If [Adjacent clusters] is not checked, the same distance matrix used for multi-dimensional scaling is used for cluster analysis (using the Ward method). In this case, adjacent words are not always classified into the same cluster, creating "enclaves." This option enables you to apply different methods for multi-

dimensional scaling and cluster analysis, deepening your understanding of the data through comparison of their results. This is because "by combining cluster analysis that displays the results with a discrete structure and MDS that shows the results with a continuous structure in a multi-dimensional space, we can extract information contained in data from different aspects" (岡太 2002: 169).

■**Common operations** Given that plotting too many words at once on a scatter diagram results in a screen filled with black, it is necessary to filter out some extracted words using the same method described in section A.5.8 for [Correspondence Analysis]. As for the [Correspondence Analysis] command, caution should be taken to set an appropriate "Unit". Clicking the [Config] button after analysis will allow you to change options, such as "Method", "Distance", "Dimensions", "Bubble plot", "Indicate clusters by different colors", "Font size" and "Plot size". The procedure for saving the plot is the same as for the [TF-DF Plot] command (see section A.5.4). The options [Bubble plot] and [Indicate clusters by different colors] are only valid when creating one- and two-dimensional maps, but not three-dimensional maps.

A.5.10 [Hierarchical Cluster Analysis] of words

Overview

This command enables you to find and analyze which combinations or groups of words have similar appearance patterns using hierarchical cluster analysis. The analysis results are displayed in a dendrogram, as shown in the example in Figure A.31(a). This command produces results that are easier to interpret than those created using the [Multi-Dimensional Scaling] command (see section A.5.9). This analysis also uses the matrix generated with the [Export Document-Word Matrix] command, with the variables for positions and document lengths removed.

Options

■**Unique options** Specifying the number of clusters under the entry labeled "N of clusters" on the right, divides the words into this specified number of clusters. The results are shown on a dendrogram, using different colors or rectangles to indicate divisions. You can choose from three different clustering methods: "Ward", "Average", and "Complete". The bars on the left side of the colored dendrogram shown in Figure A.31(a) indicate the TF of each word.

Clicking [Agglomeration] at the bottom of the dendrogram shown in Figure A.31(a), opens the plot shown in Figure A.31(b). On this plot, using the changes in the distance coefficient, you can consider the agglomeration process in detail. Clicking the buttons [<< first 50], [all], or [last 50>>] to the right of "Stages" on the bottom left of the plot window shown in Figure A.31(b) changes the plot display so you can view the agglomeration at these three different stages.

■**Common operations** Similar to other analysis windows, here you can also select a distance coefficient, as described in detail under the [Multi-Dimensional Scaling] command (see section A.5.9). Given that analyzing too many words at once can make the results difficult to understand, it may be necessary to filter out words, as described for the [Correspondence Analysis] command (see section A.5.8). As for the [Correspondence Analysis] command, caution should be taken to set an appropriate "Unit" in the upper left corner. Likewise, options such as "Method", "Distance", "N of clusters", "Font size", and "Plot size" can be changed by clicking the [Config] button on the bottom left of the dendrogram. The procedure for saving the plots is the same as for the [TF-DF Plot] command (see section A.5.4).
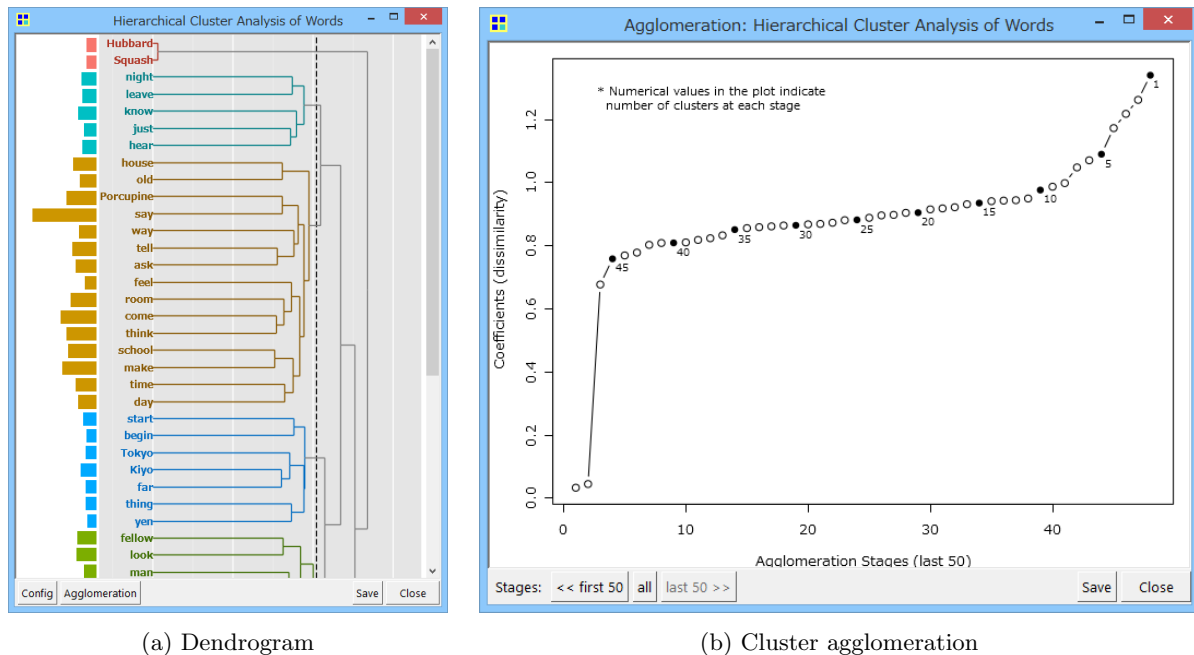
(a) Dendrogram   (b) Cluster agglomeration

Figure A.31   Windows related to a Hierarchical Cluster Analysis
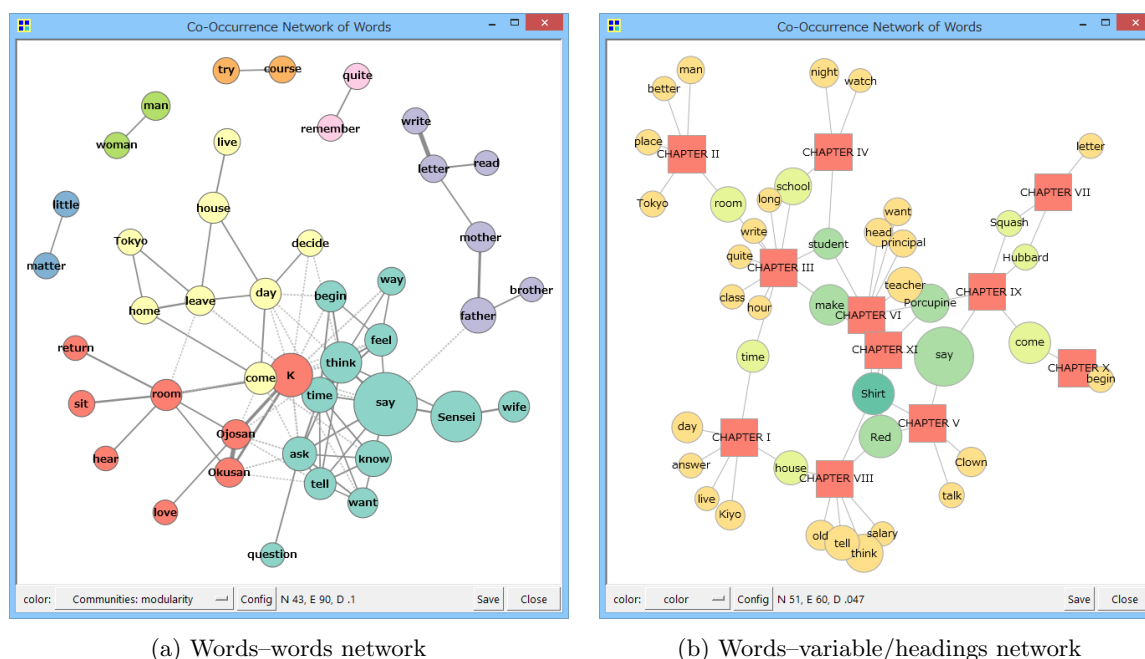
A.5.11   [Co-Occurrence Network] of words

Overview

This command enables you to draw a network diagram that shows the words with similar appearance patterns, i.e., with high degrees of co-occurrence, connected by lines (edges), as shown in Figure A.32(a). Given that words are connected with lines, it may be easier to understand their co-occurrence structures, compared with plots generated with the [Multi-Dimensional Scaling] command (see section A.5.9). The co-occurrence network has been used successfully as an analysis technique since the early stages of content analysis (Osgood 1959). More recently, the use of network analysis indices have been employed (Danowski 1993).

   This command shows the association between words and variables/headings, in addition to the associations between words. For example, Figure A.32(b) presents analyses from the novel Botchan by Soseki Natsume, showing the headings of chapters ("Chapter I", "Chapter II", etc.) and frequently appearing words, and how they connect with each other.

   When using this command, we first recommend using lines to represent only a limited number of strong co-occurrences. Once your first analysis reveals an overall picture, you can add lines (edges) representing relatively weaker co-occurrences.

   To determine word locations, this command uses the method developed by T. M. J. Fruchterman & E. M. Reingold (1991) for drawing a [words–words] network, and the method developed by T. Kamada & S. Kawai (1988) for drawing a [words–variables/headings] network. These methods arrange words so that the resulting network is easy to read. Hence, unlike for results of multi-dimensional scaling, words plotted close to each other do not always mean they have a strong co-occurrence. Instead, whether or not the words are connected with lines (edges) is significant. Although words may be plotted close to each other, if they are not connected with lines (edges), there is no strong co-occurrence. To measure the degree of co-occurrence, this command also uses the matrix output from the [Export Document-Word Matrix] command (in menu Tools > Documents, see section A.6.7), with the variables that indicate

(a) Words–words network

(b) Words–variable/headings network

Figure A.32   Examples of co-occurrence networks

positions and document lengths removed.

The user may also like to consult further useful information on this command provided in a thread in the discussion forum, available at: https://sourceforge.net/p/khc/discussion/222396/thread/2da0ff02/

Options

■**Filtering edges**   Plotting all co-occurrence information may simply fill the screen with lines. To prevent this, select and display only edges representing strong co-occurrences. To determine edge strength, Jaccard coefficients are calculated for all possible combinations of target words. Selecting the [Top (number)] option under "Filter edges" on the right of the window shown in Figure A.33, will draw only a specified number of edges with the strongest Jaccard coefficients. If the [Jaccard coef.] option is selected, then all edges with Jaccard coefficients equal to or greater than a specified threshold value will be drawn. Even though a word (node) may be the target of analysis, if it has no edges that satisfy the above criteria, then it will not be shown on the diagram.

The numbers of words (N: nodes) and co-occurrences (E: edges) shown on the diagram are given at the bottom of the result window, to the right of the [Config] button, as shown in Figure A.32(b). Here, "N 51, E 90, D .071" indicates that 51 words were drawn as nodes, 90 co-occurrences as edges, while the density (D) according to social network analysis was 0.71. The density is defined as the number of co-occurrences that are actually drawn divided by the number of co-occurrences (edges) that could exist. Additionally, if you place the cursor over the values "N 51, E 90, D .071" for approximately 1 s, more detailed information pops up, in this case "Nodes 51 (71), Edges 90 (206), Density .071, Min. Jaccard .225". The values in parentheses show the numbers of nodes and edges contained in the input data, giving the user a clearer idea of the amount of information withheld from the diagram by selecting only strong co-occurrences. "Min. Jaccard" represents the Jaccard coefficient threshold value for this diagram, i.e., the value of the weakest co-occurrence in the diagram. Thus, only co-occurrences with a Jaccard coefficient equal to or greater than this threshold value appear in this diagram.

■**Degree of co-occurrence and line (edge) thickness**   Selecting the option [Thicker lines for stronger edges] on the right side of the Options window shown in Figure A.33 changes the thickness
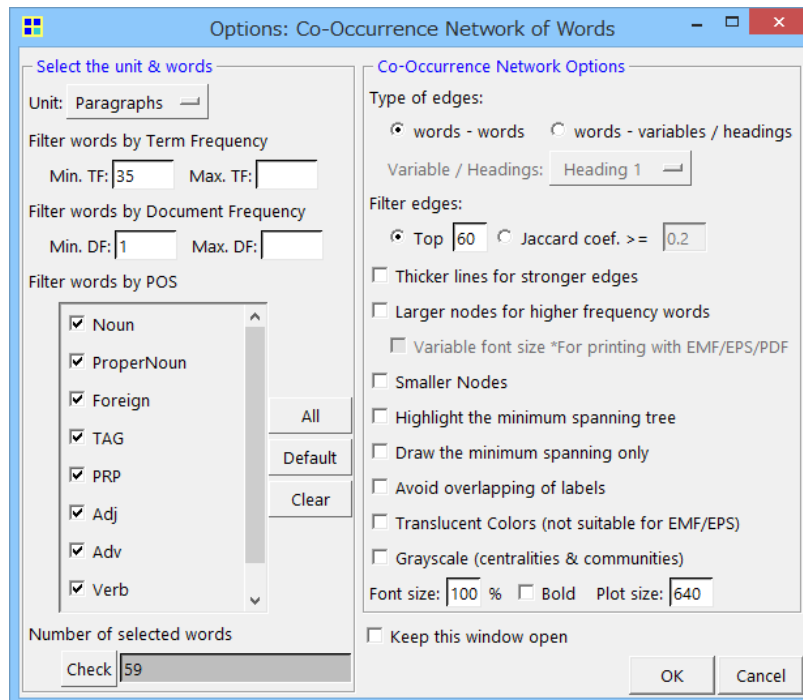
Figure A.33   Options for a co-occurrence network

of each line (edge) according to the value of its Jaccard coefficient. Figure A.32(a) was created using this option. For example, the edge that connects the words "write" and "letter" is thicker than most others, indicating that these two words have a stronger co-occurrence in the target file, compared with other words shown here.

■**Frequency and word (node) size**   Selecting the option [Larger nodes for higher frequency words] on the right side of the Options window shown in Figure A.33 changes the size of the circles representing each word (node), according to their appearance frequency.

   If [Variable font size] also is checked, the font size of the words changes, as well as their circle size. However, in plots displayed both on the screen or saved in PNG format, font size cannot be increased/decreased linearly for technical reasons. Thus, if the appearance frequency of the words exceeds a certain value, the size and thickness of its font increases drastically, making the plot unbalanced. Therefore, the [Variable font size] option should only be used, when saving a plot in EMF, EPS or PDF format for printing or exporting to LaTeX or Microsoft Word.

   In contrast, selecting the [Smaller Nodes] option reduces the sizes of all circles, putting their labels on the upper right of each circle, as shown in Figure A.26. Although plotting larger circles enables you to distinguish colors more easily, it may also obscure lines behind circles in some cases. We recommend using these options ([Larger nodes for higher frequency words] or [Smaller Nodes]) based on whether color or line visibility is more important in your plot type.

■**Colors**   On the diagram showing the "words–words" network (Figure A.32(a)), there are six different color coding schemes to choose from for words (nodes). The first three schemes are based on "centrality" from social network analysis, i.e., on how central the role each word plays in the network. Here, circles representing nodes are colored with light blue through white to pink, reflecting their ascending degree of centrality. The centrality types in this option are: [Centrality: betweenness], [Centrality: degree], and [Centrality: eigenvector](Bonacich's centrality). For more detailed information on each centrality

measure, please consult textbooks on social network analysis.

The other three color schemes are based on "communities" (sub-graphs) that are used to represent parts of the network that are more closely associated with each other through color coding.

Here, the available methods include: [Communities: betweenness] described by M. Newman & M. Girvan (2004); [Communities: random walks] by P. Pons & M. Latapy (2005); and [Communities: modularity], by A. Clauset et al. (2004). In our color coding scheme, a word on a white background enclosed in a black circle is independent, i.e., not grouped with other words. Given a maximum of 12 colors available for color coding, the 13th and onwards groups are indicated with a white background enclosed in blue circles. In these plots, words in the same community (sub-graph) are connected with solid lines, while those in different communities (sub-graphs) are connected with dashed lines.

The last option for color is [None], which is suitable for black-and-white printing.

In KH Coder, color coding provides subsidiary information for interpreting a diagram. Thus, colors may change, when you select a different community detection method or centrality calculation method. In principle, this is similar to cluster analysis, since clustering may also change when you select a different distance measure or clustering method. Hence, it is recommended not to read too much into color coding results.

■**Minimum spanning tree** For cases, where many edges connect node words, it is helpful to have an option to indicate which edges are most important in a network. Here, we provide an option for highlighting edges based on whether or not they are part of the minimum spanning tree. More specifically, selecting the option [Highlight the minimum spanning tree] on the right of the window shown in Figure A.33, shows the minimum spanning tree based on the strength of co-occurrence using the Prim method, and draws all edges that form this minimum spanning tree with darker thicker lines.

Checking the [Draw the minimum spanning tree only] option reduces the diagram to a simple network. In this function, edges other those forming the minimum spanning tree are deleted, once networks are drawn in accordance with the threshold value given in [Top] under [Filter edges]. Hence, the number of edges drawn will be less than that specified in the Options window.

■**Avoiding overlapping of labels** Word labels that are close to each other in the network can overlap and be difficult to read. Selecting [Avoid overlapping of labels] on the right in the Options window (Figure A.33) adjusts label positions so they do not overlap. Such fine adjustment uses the "wordlayout" function included in the "wordcloud" package of R. This command basically tries to avoid overlapping with only minimal word movement. However, when words are densely packed, this command may move them long distances.

■**Common operations** Since analyzing too many words at once can make the diagram too packed with items, and filtering may be required, as described in the [Correspondence Analysis] command (in menu Tools > Words, see section A.5.8). Again, caution should be exercised in setting an appropriate "Unit" in the upper left corner. Likewise, options such as "Filter edges", "Thicker lines for stronger edges", "Larger nodes for higher frequency words", "Font size" and "Plot size" can all be accessed by clicking the [Config] button at the bottom of the diagram. The procedure for saving diagrams is the same as for the [TF-DF Plot] command (in menu Tools > Words > Descriptive Stats; described under section A.5.4).

A.5.12 [Self-Organizing Map] of words

Overview

This command enables you to explore associations between words by creating a self-organizing map. To create a self-organizing map in KH Coder, the matrix output from the [Export Document-Word Matrix]

command is used, with the variables for positions and document lengths removed. However, given that creating a self-organizing map uses Euclidean distances, standardization is carried out on each word, as described for calculating Euclidean distances under[Multi-Dimensional Scaling]. It should be noted that using default settings may take a very long time to generate a self-organizing map. On a Core i7 CPU PC, it took about 5 min to process approximately 70 words × 100 documents, but 64 min to process approximately 70 words × 1,200 documents.
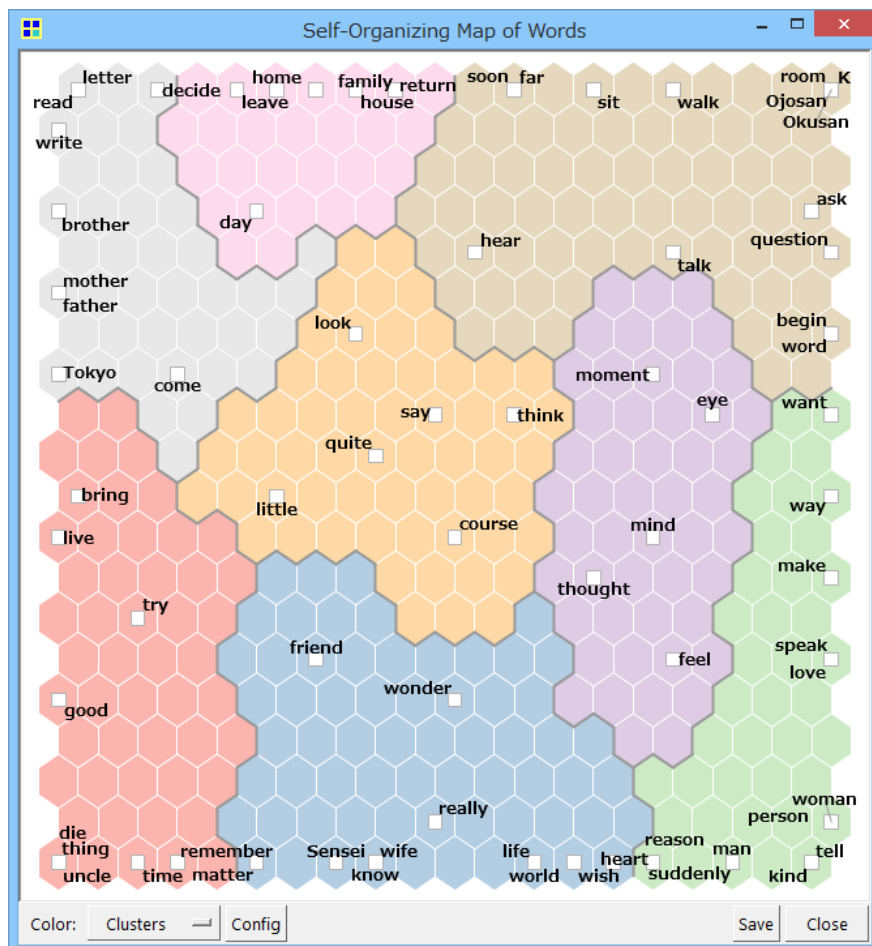
Figure A.34   Example of a self-organizing map of words

Options

■**The number of nodes (per side)**   This option specifies the size of the map. The default size is 20 nodes. Using this default value, a map of 20 (vertical) nodes × 20 (horizontal) nodes, with a total of 400 nodes is created. Since the processing time increases significantly with the number of nodes, a realistic number of nodes should be between 20 and 30.

■**The number of steps**   In KH Coder, the training steps to generate a self-organizing map are divided into two phases: an ordering phase for defining a rough sequence, and a convergence phase for fine tuning. The default settings for the numbers of steps are 1,000 for the first phase; and "Auto" for the second phase. The "Auto" option gives the number of steps for the second phase as the number of nodes for the entire map multiplied by 500. Default values are defined based on T. Kohonen (2001: 112).

■**Clustering nodes**   Clustering of nodes can be performed and indicated using color-coding, within map borderlines to provide supplementary information for interpreting the self-organizing map. Because the vector (codebook vector) for each node is classified using the Ward method, groups of nodes adjacent to each other are often classified as a single cluster. Hence, cluster "enclaves" rarely occur.

■**Plot types**   Hexagonal nodes are always used for the calculation of the self-organizing map. They cannot be switched to rectangular ones. However, when drawing a map, you can choose to arrange hexagons seamlessly, as shown in Figure A.34, or to use rectangles. These options only affect the appearance of the plot.

■**Colors**   At the bottom left of the map, you can select among several coloring options. If you checked [Clustering nodes], then you can choose the [Cluster] option that color-codes nodes according to their clusters. In the [Grayscale] option, no coloring is performed, regardless of whether or not the nodes are clustered. The [Frequency] option displays nodes with more words with darker colors. Lastly, the [U-Matrix] option colors nodes using a gradient from light blue through white to pink to reflect increasing Euclidean distance between adjacent nodes. Plotting is based on the median Euclidean distance calculated from six vectors of six adjacent nodes. Thus, a pink line between nodes means that the appearance patterns of these words differ; thus, it can be seen as a "gorge" that divides the clusters. In contrast, a group of blue nodes indicates that these words have a similar appearance patterns and form a cluster. In this way, cluster structures can be identified from a U-matrix network diagram. It is also possible for the user to determine whether an appropriate number of clusters was specified by comparing the results of clustering using the Ward versus U-matrix methods.

■**Common operations**   When a diagram is too packed with information, it may be necessary to filter out words, as described in the [Correspondence Analysis] command. Caution should be taken to set an appropriate "Unit" size for this operation. Likewise, options, such as "Number of clusters", "Plot", "Font size" and "Plot size" can be changed by clicking the [Config] button at the bottom of map. The procedure for saving the map is the same as for the [TF-DF Plot] command (section A.5.4).

However, it is important to note that the options "Number of nodes (per side)" and "Number of steps" cannot simply be changed by clicking the [Config] button, once results are displayed. Since it takes time to process the training steps for the self-organizing map, clicking the [Config] button enables you only to re-plot and customize your self-organizing map, using all processed data. To change the input value for "Number of nodes (per side)" or "Number of steps", it is necessary to re-select the [Self-Organizing Map] command from the menu, enter new values, and then re-process the training steps.

A.6   [Tools] > [Documents] menu

A.6.1   Search Documents

Overview

This command enables you to search documents using various conditions, such as "includes a specific word." It is also possible to search documents with a specific code. This command is useful for checking whether the coding rules you created perform as expected. Using a sort index, such as tf*idf, you can retrieve documents that represent a specific code content or association between codes.

Executing this command opens the Search Documents window like the one shown in Figure A.35. Here, you can specify the unit size of a single document to be searched. If there are more than 200 search results, then only 200 are displayed at a time. Additional results can be viewed by clicking [N200] or [P200] to browse the next or previous set of 200 results, respectively. Each result shows only the first 40 characters of the matched document. To read a given document line in detail, double-click this line,
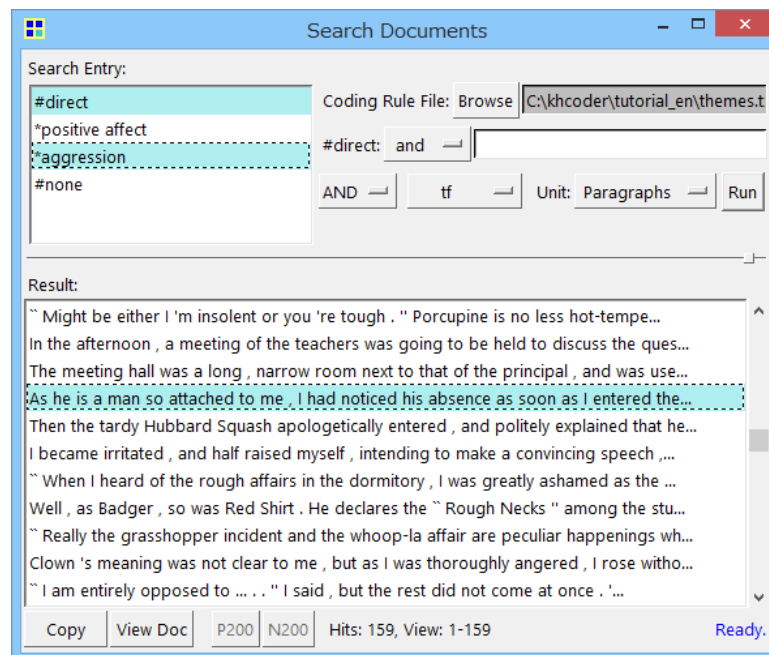
Figure A.35   Example of a Search Documents window

or select the line and click the [View Doc] button at the bottom. This opens the Document window shown in Figure A.36(a), displaying the entire document.

Options

■**Coding rules**   Clicking [Browse] to the right of "Coding Rule File" in the upper right opens a dialog box. Here, you can select a coding rule file that lists codes that will appear under "Search Entry" on the left. This enables you to search for documents with one of the displayed codes by clicking [Run] on the upper right. In the Windows version, you can also open a coding rule file simply by dragging and dropping the file onto the [Browse] field.

You can also select several codes at once and search using AND or OR operators. As for other Windows applications, more than one code can be selected by clicking codes, while depressing the [Ctrl] or [Shift] key. When searching for multiple codes, the menu button labeled "AND" (the default) becomes active; clicking this menu button and choosing "AND" or "OR" allows the user to do an AND or OR search with multiple codes.

■**The "#direct" Code**   The "#direct" in the "Search Entry" list shown in Figure A.35 can be used to enter a search word directly, without creating a coding rule file. In a direct entry search, three modes are available: "and", "or", and "code". You choose the mode by clicking the menu button to the right of "#direct". Clearly, "and" performs an AND search, which retrieves the documents that contain all the specified words, while "or" performs an OR search, which retrieves documents that contain at least one of the specified words. To specify several words at once in this input box, just enter the desired words delimited by spaces. When selecting "code", you can search documents by entering the conditional part of the coding rule. This mode allows you to use all the search conditions available under the coding rule options.

We found the direct entry search very useful for temporarily adding conditions to codes read from the coding rule file. For example, in the "Search Entry" code list you can choose "#direct" plus a specific code from the coding rule file, then also enter a word in the "#direct" input box, and perform an AND

search on all these conditions. This function can be especially useful for creating and testing new coding rules.

Sort order

Four sorting options are available for ordering your search results: "no sort", "tf", "tf*idf", and "tf/idf". You can change result order by clicking the menu button showing "tf" shown on Figure A.35 (note: the default is "no sort"). Selecting "no sort" arranges the documents in the order they appear in the target file. Using "tf", "tf*idf", or "tf/idf" arrange the documents depending on how the code or the association between codes is represented in the text data. In descending order, the most typical document comes first. The meaning of "typical" differs among these three sort options, and you can choose the one that best suits your purpose.

First, the function *tf(d, t)* is obtained by dividing the frequency of the word t in document d by its length, where the length of document d is the number of words contained in the document, i.e., the number of morphemes.

tf(d, t) = Frequency of the word t in the document d / Length of the document d

Thus, the more frequently word t appears in the document, or the shorter the document, the larger the value for *tf(d, t)*. Similarly, the value of *idf(t)* is smaller, if there are more documents containing the word t, and larger, if there are fewer such documents. The *idf(t)* value is calculated in the formula shown below, where the log represents the base 10 logarithm.

idf(t) = log (number of all documents) / (number of documents containing word t)

■[**tf**]　Choosing this sort order calculates a *g(d, c)* value, which gives the degree of typicality with which the document d represents the code c, for the case where the conditional part of the coding rule contains n words. This value is calculated, as follows:

$$g(d,c) = \sum_{i=1}^{n} tf(d, t_i)$$

For example, consider the following coding rule:

```
1  *Friendship
2  friend | pal | mate | friendship | companion
```

Using *tf(d, t)*, the more words in a document like "friend" and "pal", the larger the value *g(d, c)*. If two documents contain the same number of such words, then the shorter document will have a larger *tf*. Therefore, the shorter the document, and the more target words in the document, the greater the *g(d, c)* value. In this way, these high-valued documents are considered to more typically represent the code "*Friendship".

■[**tf/idf**]　In *tf*, each word has the same weight for calculating *g(d, c)* values. In contrast, in *tf/idf*, the weight varies among words. Specifically, words that appear in a larger number of documents are considered more typical of the code, and are given more weight. Here, the value of *g(d, c)* is represented by:

$$g(d,c) = \sum_{i=1}^{n} \frac{tf(d, t_i)}{idf(t_i)}$$

Thus, in the example of the code "*Friendship" above, if the word "friend" appears in more documents than "pal", then "friend" is judged to be more representative of the code. Hence, if two documents have the same length, yet "friend" appears once in one document and "pal" appears once in the other document, then the document that contains "friend" will have a larger *g(d, c)*, because "friend" more typically represents the code "*Friendship".

■**[tf*idf]**   In calculating the *g(d, c)* value for *tf*idf*, weight is assigned to the *tf* value in the manner opposite to *tf/idf*.   Hence, the words that appear in a smaller number of documents are considered significant, because they show the specificity of the code. Here, the *g(d, c)* value is given by:

$$g(d, c) = \sum_{i=1}^{n} tf(d, t_i) idf(t_i)$$

Document display windows

Examples of the document display windows are shown in Figure A.36.

(a) Document screen   (b) Highlight Words/Strings screen
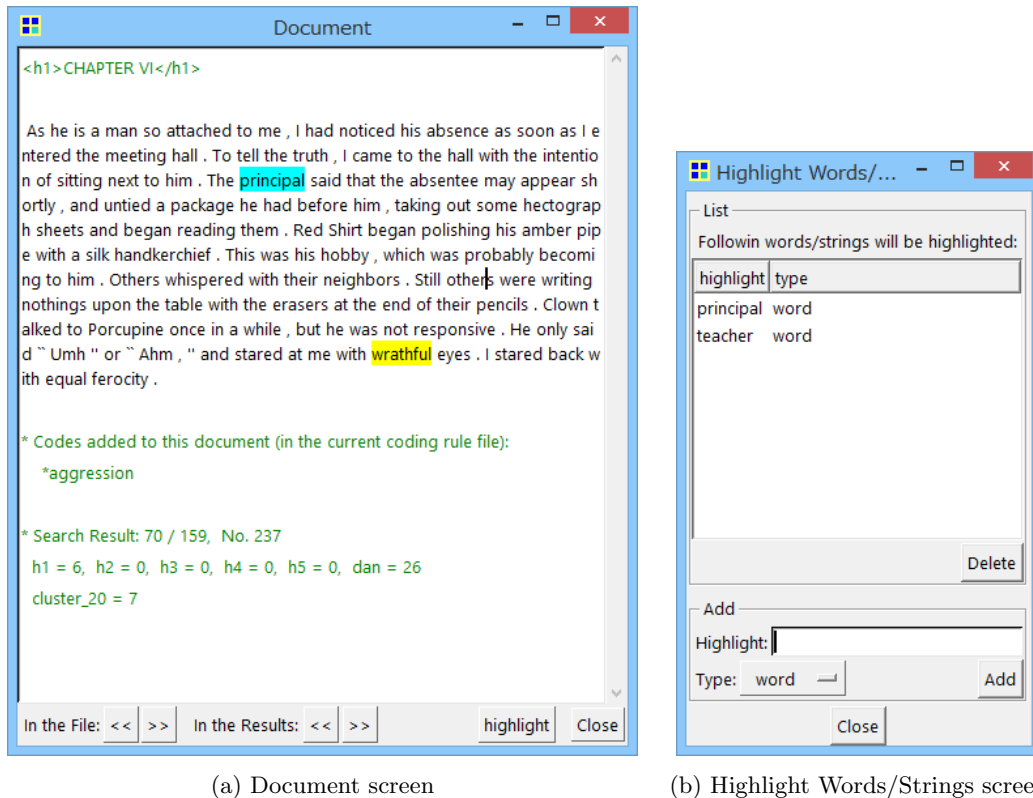
Figure A.36   Examples of document display windows

■**Highlight**   In Figure A.36(a), search words are highlighted with yellow, while other words that are used for coding appear with blue highlighting, and HTML tags appear green. Highlighting can be set to pick up word occurrences in a given text. For example, when you are analyzing newspaper articles describing a specific theme, it may be convenient to highlight words associated with this theme. To do this, click the [Highlight] button, which opens the Highlight Words/Strings window shown in Figure A.36(b). Here, you can specify the words you want highlighted. If a word to be highlighted also was extracted by KH Coder, you can add this word to what appears under [List], by choosing "word" in the menu toggle to the right of "Type". If the word was not extracted by KH Coder, you can add it as a "string", using the same menu.

■**Additional information**   Any additional information about the document currently displayed is indicated in green. Headings at levels higher than the selected document, are shown on the first display line, while in the lowest part of the window, "codes added to this document (in the current coding rule file)" are listed. The sequential number of the currently displayed document in the search results and the

sequential number of the document in the target file are indicated, as well as more detailed information on position and variable values, similar to that used in the [Export Document-Word Matrix] command (see section A.8.1).

■**Other operations**   If you want to check the previous or next context of a word in the same file, click the [<<] or [>>] buttons, respectively, found to the right of "In the File" on the bottom left window corner. Likewise, to navigate backwards and forwards through the search results, click the [<<] and [>>] buttons, to the right of "In the Results". In this window, the following shortcut keys are available:

- [Shift] + [PageUp] Displays the previous document in the file
- [Shift] + [PageDown] Displays the next document in the file
- [Ctrl] + [PageUp] Displays the previous search result
- [Ctrl] + [PageDown] Displays the next search result

A.6.2   [Cluster Analysis] of documents

Overview

This command performs cluster analysis on a document-level and saves the results as variables (see section A.8.1). Using this command, you can find which documents contain similar words. It can also generate a dendrogram, as in Figure A.31(a), which clearly shows which documents are similar to each other. This command enables you to analyze the contents of each cluster, using the list of words characteristic to a cluster. This analysis uses the table generated with [Export Document-Word Matrix] (see section A.6.7), but with the variables for position and document length removed.

   Given that the analysis is performed using R, the size of the data file to be processed is limited. For example, analyzing a target file that contains 5,000 documents, with around 5,000 types of words used for the analysis, requires 1 GB of free physical memory. Larger data files may cause an error during execution of this command; in such cases, we recommend performing a random sampling of the data to reduce the data fed to KH Coder. If it is necessary to analyze larger amounts of data, a sufficient amount of physical memory and the 64-bit version of R should be available. However, these steps only partly alleviate such limitations on data size. Another alternative is to use the [Export Document-Word Matrix] function (see section A.6.7) to export the data for analysis with statistical software that can process larger amounts of data.
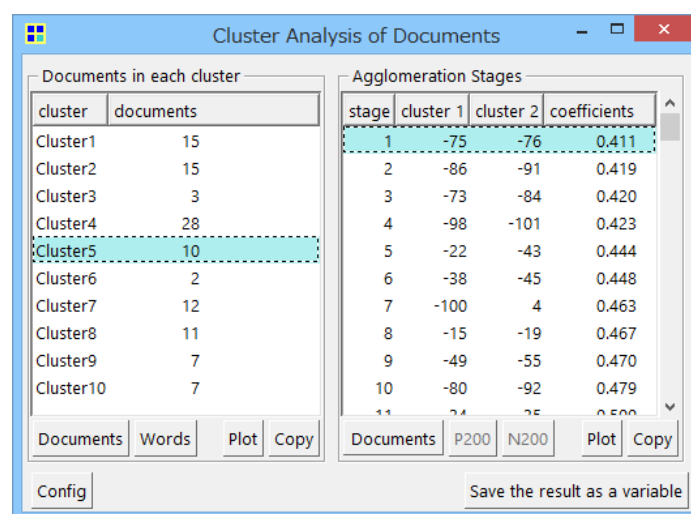

Figure A.37   Cluster analysis at a document-level

Options

■**Selecting words to analyze**   Similar to other analysis windows in KH Coder, the words used for cluster analysis can be selected using frequency and POS classification criteria. The procedure used to select words is outlined under [Correspondence Analysis] (section A.5.8).

If a smaller number of word types are selected for analysis, then it is possible that some of the documents contain no selected words at all. In such cases, KH Coder labels such a document with "n/a" and withdraws it from analysis, as indicated in the "Documents in each cluster" area on the left of the window in Figure A.37.

■**Distance coefficients and choice of cluster method**   Similar to cluster analysis for words (section A.5.10), "distance" can be calculated using [Jaccard], [Euclid], or [Cosine] methods, under the "Distance" menu. If [Euclid] or [Cosine] are selected, the calculation uses the number of occurrences per 1,000 words (adjusted frequency) rather than the number of occurrences in the documents themselves (raw frequency). Depending on which "Method" and "Distance" options are checked, you can choose [tf/idf] or [tf] to add weight to more relevant words appearing in only a few documents. Clicking the menu to the right of "Standardize" allows you to choose whether or not to standardize your data. Data may be standardized using options: [None], [By Words], and [By Document].

Four clustering methods are available under this option: [Ward], [Average], [Complete], and [CLARA]. [CLARA] (Clustering LARge Applications) enables you to process data, with a relatively large number of documents (Kaufman & Rousseeuw 1990). However, if the size of the data set is too large, problems occur in the data preparation stage. On a 32-bit operating system, for a number of documents around 100,000, the number of words should be limited to between 200 and 300. If [CLARA] is selected as the "Method", the "Distance" is fixed to "Euclid", and the "Agglomeration Stages" are not displayed (Figure A.37).

In addition, the options "Distance", "Method" and "N of clusters" can be customized using the [Config] button, after results are produced (Figure A.37).

■**Checking the cluster contents**   The left side of the Cluster Analysis of Documents window shown in Figure A.37, lists only the number of documents classified in each cluster. Selecting a certain cluster and clicking the [Documents] button below the panel, lists the documents contained in that cluster in a new Search Documents window (see section A.6.1). Instead, selecting a certain cluster and clicking the [Words] button opens the Word Association window (see section A.5.7), which shows the words that appear with higher probabilities in the documents contained in the cluster, or the words that represent the characteristics of the cluster. These operations also can be performed by double-clicking each cluster to activate the Search Documents window, or double-clicking it, while pressing the [Shift] key to activate the Word Association window. Clicking the [Plot] button displays a dendrogram like that shown in Figure A.31(a), in which it is easy to visually identify documents that are similar to each other.

■**Checking the Agglomeration Stages**   In Figure A.37, the right window side displays the "Agglomeration Stages". Numbers with a negative sign in columns "cluster 1" and "cluster 2" give the sequential ID number of the document. Clearly, in the first stage, or "stage 1", the 106th and 107th documents were agglomerated into a single cluster. Numbers without a negative sign in these columns indicate the stage number in which the cluster was agglomerated.

Double-clicking one of the lines below the "Agglomeration Stages" opens a Search Documents window (section A.6.1) that shows the documents agglomerated in that stage. You can also select a specific line (stage) in this area and click the [Documents] button on the right to display three options: "1 & 2" (same as double-clicking the line), "1 only", and "2 only". These options will open a [Search Documents] window to search documents in both "cluster 1" and "cluster 2", only "cluster 1", or only "cluster 2",

respectively.

Clicking the [Plot] button below "Agglomeration Stages" opens the Agglomeration: Cluster Analysis of Documents window shown in Figure A.31(b). Plotting the changes in the distance coefficient during cluster agglomeration provides a basis for determining the number of clusters, and examining the agglomeration stages in detail. Clicking the buttons [<< first 50], [all], and [last 50 >>] changes the plot display so you can view different stages: the entire agglomeration process ([all]), or only the first 50 stages or last 50 stages, in which the most significant changes tend to occur.

■**Saving the Results** You can save your results as variables by clicking [Save] and specifying a variable name. If you do not save the results in this step, the results will be deleted, when the results window is closed.

A.6.3 [Naive Bayes Classifier] > [Build a Model from a Variable]

About the Classifier

In the "Naive Bayes Classifier" menu, you can choose this function. Once a number of documents are classified manually, this function learns classification criteria to automatically classify other documents. This learning process uses the Naive Bayes model, which is designed to perform exclusive classification or categorization.

Alternatively, given that a single document can refer to several things, you can perform a process similar to coding in KH Coder, using repetitive learning and automatic classification based on categorization, such as "item A referred to" and "item A not referred to" on several items. In this way, KH Coder can perform the processing required to determine whether the document refers to each item using this classifier.

Using this function, a researcher can analyze data from a viewpoint different from creating coding rules. For items and concepts like "human death" illustrated in the tutorial on the novel Kokoro by Soseki Natsume, coding rules can easily be created. However, for broader expressions, with greater textural variety, it can be difficult to create coding rules, especially of the type "if the document contains these expressions, item A can be regarded as appearing in the document." Even in such cases, if a sufficient number of classification examples are presented, it is possible to attempt automatic classification using this function.

Given that coding rules define classification criteria, this function can be regarded as automatically generating a coding rule (or set of classification criteria) based on the set of manual examples. Hence, it is important that the contents of the learned classification criteria can be verified by the analyst and that they can be explicitly demonstrated to work to third persons, as with the coding rules. Therefore, KH Coder uses the Naive Bayes model as its method of machine learning, because its content on classification criteria is in human readable format. It also provides functions for browsing and checking these learning contents.

About the Naive Bayes Model

According to Bayes' theorem, a conditional probability can be represented as:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

where $p(x|y)$ is the probability of the occurrence of x under the condition y.

It is not possible to directly calculate $p(C|W)$, i.e., the probability that a document that contains a series of words W belongs to category C. However, using the above theorem, $p(C|W)$ can be represented as:

$$p(C|W) = \frac{p(W|C)p(C)}{p(W)}$$

Transforming the formula in this way makes it easier to estimate the value of $p(C|W)$. The Naive Bayes model used by KH Coder calculates this value for all categories, and classifies the documents into the category with the largest value. Clearly, the common denominator $p(W)$ can be omitted because it is common to all categories, and only the numerator is calculated and used as the score for each category. When n types of words are contained, $p(W|C)$ is calculated as follows:

$$p(W|C) = p(w_1|C)p(w_2|C) \ \cdots \ p(w_n|C)$$

KH Coder uses a Perl module called "Algorithm::NaiveBayes" created by Ken Williams. The following descriptions are based on documents attached to this module.

The calculation does not actually obtain the value of the numerator, $p(W|C) \ p(C)$, but rather its natural logarithm. This is because calculating the product of a large number of probabilities results in a value close to zero, which can cause a problem of significant figures in a floating-point calculation. To solve this problem, the following a logarithmic property can be used.

$$\log xy = \log x + \log y$$

Thus, the value of $p(W|C) \ p(C)$ can be calculated using:

$$\log p(W|C)p(C) = \log p(w_1|C) + \log p(w_2|C) + \cdots + \log p(w_n|C) + \log p(C)$$

The logarithmic values $\log p(wi|C)$ can be determined in the learning phase. Their addition promotes a faster and more accurate calculation than multiplication in the classification phase.

In short, classification using the Naive Bayes Model involves the following steps. In the learning phase, the value of $\log p(wi|C)$ is calculated from the classification models. Log $p(wi|C)$ reflects the score that should be added to category C, when the word i appears once in the document during automatic classification. It follows naturally from the definition that the more frequently the word i appears in category C in the manual classification examples, the larger the value of $\log p(wi|C)$. This model is based on a simple concept: if a word appears many times in category C in the manual classification examples, then the documents that contain more instances of this word are more likely to be classified to category C as well.

The options described in the sections below, including [View a Model File] and [View a Classification Log], display the adjusted value of $\log p(wi|C)$ as the score of the word by category and the adjusted value of $\log p(W|C) \ p(C)$ as the score of each category. Since $\log p(wi|C)$ has a negative value, the value is converted to a positive one, by adding a value such that the minimum value becomes 0. The sum of the converted values is indicated as the adjusted $\log p(W|C) \ p(C)$. This converted value of $\log p(C)$ is shown under the name "prior probability".

As described above, the positive scores are different from those actually used for the calculation. However, these are easier to understand compared with a series of negative numbers. Since the conversion value is the same for all categories, the calculation and classification mirror the results for the actual values. Thus, each score reflects the classification process, such that, if the score of word "A" for category "Alpha" is high, then the frequent appearance of word "A" leads the document to be classified into category "Alpha".

Command overview

This command performs learning based on manual classification examples or teaching signals. The results will be saved to a file with the extension "*.knb". These classification examples need to be input to KH

Coder as variables. If the value of the variable or value label is "MISSING", ".", or "欠損値", then it is treated as a missing value and is not used for training.

The saved model file also can be used for other projects. Hence, you can use the training results developed in project A to classify documents in project B. Moreover, you can perform further training in project C, and append these results to the existing model file.
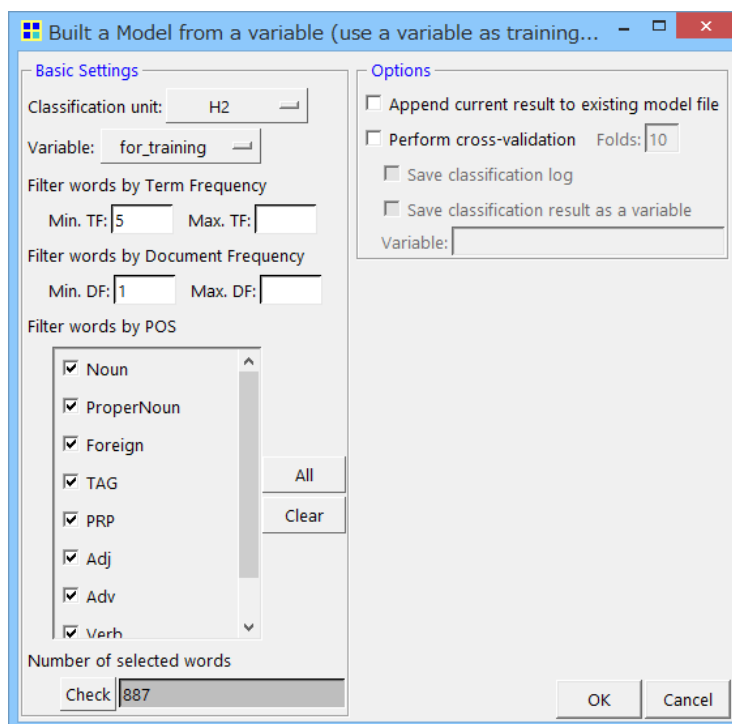

Figure A.38 Build a Model from a variable window

Options

■**"Classification Unit" and "Variable"**   In the command window for this function, you are able to specify the unit that indicates the range of a single document (section A.2.1). It is also necessary to select the variable from which to learn the classification criteria.

■**Filtering words**   The words used for training can be selected according to the three criteria: "TF", "DF", and "POS" (defined in section A.5.8). Before actually creating the model file, you can see how many words pass the screening options you have set using the [Check] button on the bottom left of this window, as shown in Figure A.38.

■**Appending results to an existing model file**   Selecting [Append current result to existing model file] in the upper right corner of the window appends the training contents in this particular cycle to an existing model file, ending with "*.knb". This is a convenient option for generating a training file from several projects.

■**Cross-validation**   Cross-validation is a testing method that withholds some documents from the training exercise so as to trial the automatic classification on those documents, i.e., the documents not used for training become the first test subjects for automatic classification. With cross-validation, you can get a sense of how accurate the automatic classification will be.

The procedure involves dividing the data into n groups, where all but one group are fed to training; this remaining group is tested to establish whether the function accurately classifies documents. This

training and testing process is repeated n times, until all data are tested. The value n can be specified in the input box to the right of "Folds", shown in the upper right of the window in Figure A.38.

Executing cross-validation will create a confusion matrix like the one shown in Table A.7, which can be saved to CSV format. The confusion matrix displays the results of cross-tabulation, with the values of the variable used as examples or correct classes, based on the results of automatic classification. Table A.7 indicates that four documents belonging to the "part 2" category are incorrectly classified to the "part 3" category. At the bottom of the table, the number of correctly classified documents, percentages, as well as the kappa statistic showing the degree of correctness are listed. The kappa value uses 0 (zero) for a correctness ratio equal to that attained through random classification, and 1 for all data correct.

Table A.7   Results of Cross-Validation

		automatic classification		
		part 1	part 2	part 3
	part 1	54	0	2
correct categories	part 2	0	32	4
	part 3	0	0	18

correct classifications 104 / 110 (94.5%)

Kappa = 0.912

The "Save classification log" option saves the details of the test classification in a log file with the extension "*.nbl". The file includes information on why each document was classified to a specific category. The contents of the log file can be browsed using the [View a Classification Log] command (see section A.6.6).

The [Save classification result as a variable] option saves the category selected as a result of the test classification, and whether that category was correct or not. Variables are created with "-class" and "-is_correct" appended to the end of the variable name supecified in the window shown in Figure A.38. Using these variables, you can easily identify, which documents failed to be classified.

A.6.4   [Naive Bayes Classifier] > [Classify Documents using a Model]

Overview

This command automatically classifies documents using learning results from manually set classification examples. Before using this command, you need to create a model file, using the [Build a Model from a Variable] function described above (section A.6.3). Results of an automatic classification are saved as a variable.
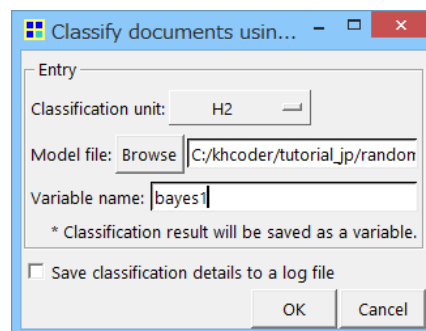

Figure A.39   Classify documents using a model file

Options

Options are set using the window shown in Figure A.39. The user firstly needs to specify a "Classification unit" that indicates the range of a single document (see section A.2.1). The model file to be used can be selected using the [Browse] button. Once the name of a variable to save the classification results is given, the automatic classification can be started.

[Save classification details to a log file] saves detailed information describing why each document was classified in a specific category, as well as the classification results in a log file. The contents of this log file can be viewed using the [View a Classification Log] command (see section A.6.6).

A.6.5 [Naive Bayes Classifier] > [View a Model File]

Overview

This command enables you to check the contents of the model file created with the [Build a Model from a Variable] command (section A.6.3). Such information allows you to view what classification criteria were obtained from the classification examples provided with the variables.

	Part 3	Part 1	Part 2	Variance	Part 3 (%)	Part 1 (%)	Part 2 (%)
Sensei-ProperNoun	0.00	5.04	3.34	4.39	0.00	60.15	39.85
K-ProperNoun	4.52	0.27	0.70	3.65	82.31	4.93	12.77
Ojosan-ProperNoun	3.91	0.27	0.70	2.64	80.10	5.54	14.36
Okusan-ProperNoun	3.76	0.27	0.70	2.41	79.47	5.72	14.82
doctor-Noun	0.00	0.96	3.10	1.68	0.00	23.72	76.28
brother-Noun	0.69	0.27	3.19	1.66	16.70	6.52	76.78
visit-Verb	0.00	2.84	0.70	1.45	0.00	80.17	19.83
thesis-Noun	0.00	2.76	0.70	1.37	0.00	79.71	20.29
father-Noun	1.39	2.06	4.10	1.33	18.36	27.31	54.33
party-Noun	1.10	0.27	3.00	1.31	25.12	6.19	68.69

Figure A.40   Contents of the model file

Options

■**Sort Order**   Running this command and selecting [View a Model File] opens a window like the one shown in Figure A.40. In the "Info" panel on the first line the fields "Processed documents" and "Types of Words" display the number of documents and types of words used in the learning phase. In the main table, the top 500 words are listed in the specified sort order. Clicking the header of each column sorts the table into descending order based on that column's scores.

The leftmost column in the table lists the words and POS classification, with delimiting hyphens "-". Names like "Part 1", "Part 2", and "Part 3" displayed in the headings shown in Figure A.40 indicate category names. Each column lists the scores of words in that category. For example, within the "Sensei-ProperNoun" row, if the word "Sensei" appears once in a document, a score of 5.04 is added to the "Part 1" category, but "Part 2" and "Part 3" receive scores of only 3.34 and 0. These scores are summed for all processed words. The category with the highest score is selected for classification. Therefore, the more times the word "Sensei" appears, the higher the score in "Part 1" category compared to the other

categories, which increases the probability that the document is classified in "Part 1". For details on these scores, refer to [Build a Model from a Variable] (section A.6.3).

Sorting with each category's scores allows you to determine which words provide high scores to a relevant category. However, the word that provides a high score to category A may also give high scores to categories B and C. The scores given to B and C may even be higher than the score given to A. Therefore, sorting by a category's scores does not always list the words that add higher scores to the selected category than to other categories. Thus, this method does not necessarily list only the words that cause a document to be classified in the selected category.

To view the words that provide higher scores to the selected category than to other categories, sort the words using a column with "%" data, such as "Part 1 (%)", "Part 2 (%)", or "Part 3 (%)". These columns list the percentages of the score of each word in each row. For example, a value of 80 in category A indicates that 80% of the scores added to all categories are distributed to category A.

The "Variance" column shows the variance or dispersion of the scores in each category. Higher variance means larger differences in scores among categories, i.e., a larger distinction between categories. Thus, to find the words that greatly affect classification, it would be better to sort using the "Variance" column. Figure A.40 shows results sorted using the "Variance" column. Leading words such as "Sensei" (teacher), "K" (a character), and "Ojosan" (young lady) have the largest effects on classification in this example.

The label [prior probability] in the word list gives a score to each document no matter what words the document contains. These scores are calculated using the function log $p(C)$, mentioned in section A.6.3. Documents, in which the words contained in the example data do not appear, or documents that consist only of unknown words, are classified according to this [prior probability] score.

■**Search**   You can search for words using the input box at the bottom of the window. Since the POS listed in the table are also search targets, a POS can also be entered here. The search field also accepts regular expressions. Regular expressions allow you to perform a search by specifying patterns. Regular expressions used in Perl are acceptable inputs. Likewise, entering several character strings delimited with a space can be used to perform an AND search.

■**Exporting Data**   The [Copy (all)] button on the bottom right or the [Ctrl+C] keyboard shortcut copies the data currently displayed to the clipboard. The table lists only the top 500 words for the custom-selected sort order. To view all of the data, click the [Whole list] button at on the bottom right, which exports all data to CSV format as a spreadsheet, and then opens this file. On a Windows PC with Microsoft Excel installed, Excel will automatically start up to display this file.

A.6.6   [Naive Bayes Classifier] > [View a Classification Log]

Overview

Using this command enables you to open and view the contents of any log files saved with the [Classify Documents using a Model] command (section A.6.4) and the "Perform cross-validation" option of the [Build a Model from a Variable] command (section A.6.3). These log files contain detailed information on why each document was classified into a specific category.

Options

■**Selecting a document**   Running this command with a log file selected will open a Classification Details window, as shown in Figure A.41. This window focuses on a specific document, showing information on the document, including its classification. The document to view is selected previously on the Document window (Figure A.36(a)) through the [Search Documents] command (section A.6.1). Thus, classification information for the document opened in the Document window is automatically displayed on the Classification Details window, providing complementary information. We also recommend
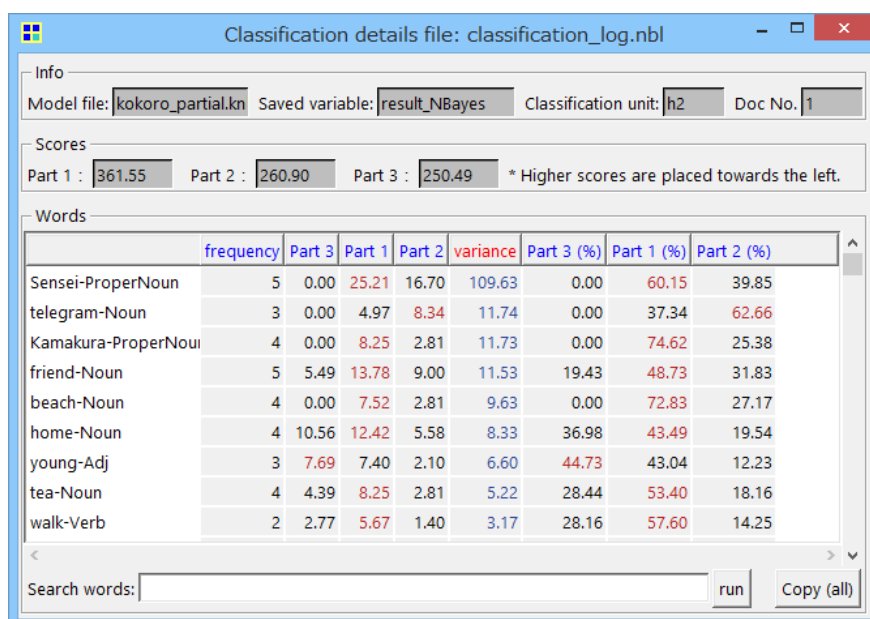
Figure A.41   Contents of the classification log

checking the original text using the Document window.

■**Display and sort order**   In Figure A.41, the "Info" panel provides information on the model file used for automatic classification (in the box to the right of "Model file") as well as the name of the variable to which the classification results were saved (in the box to the right of "Saved variable"). The "Scores" panel shows the score for each category ("Part 1", "Part 2", "Part 3") calculated for the given document. Each document is classified to the category with the highest score. Since "Part 1" has the highest score, with 361.55, the document is classified to the "Part 1" category in this example.

Details of the scores are listed in the table in the "Words" panel. The first line, starting with "Sensei-ProperNoun" indicates that this word appears five times in this document, which gives a score of 25.21 to "Part 1". "Part 2" and "Part 3" receive scores of only 16.70 and 0.00, respectively. Hence, "Sensei-ProperNoun" clearly contributes to the classification of this document to "Part 1". The sum of all scores in the "Part 1" column is 361.55, which is equal to the score shown in the "Scores" panel. This applies to other categories as well. However, since the values in the table give only two decimal places, any manual calculation with these scores will have inherent rounding errors.

The "Part 1 (%)", "Part 2 (%)", "Part 3 (%)" and "variance" columns in this table are equivalent to those headings under the [View a Model File] command (section A.6.5). To find the words that mostly influenced classification, sort the table using the "variance" column. As for the [View a Model File] command, this is achieved by clicking the header of this column.

■**Search and copy**   Similar to the [View a Model File] command (section A.6.5), you can search for a word using the input box at the bottom of the window. Clicking the [Copy (all)] button copies the data currently displayed to the clipboard.

A.6.7   [Export Document–Word Matrix]

Overview

Using this command enables you to export a tabulation matrix, showing how many times specific words appear in each document in CSV, SPSS, Tab Delimited, or variable-length CSV format (for WordMiner).

Such data can be analyzed using any statistical software, enabling more sophisticated or specialized analyses of your data, compared with analysis functions available in KH Coder.

Figure A.42   Partial View of an Exported "Document-Word" Matrix (CSV Format)

An example of data exported with this function is shown in Figure A.42. Each row (or case) represents one document, and each column (or variable) from column F onwards shows a given word's frequency. In the first row, "school" appears 14 times in the first document "CHAPTER I", while "teacher" appears only once. In addition to frequencies, the position and length of each document are described in the first several columns (variables). A description of these columns (variable) is given below:

- **h1** H1 count: Each H1 tag found increments this count by one.
- **h2** H2 count: Each H2 tag found increments this count by one. However, if a higher level heading (H1) tag is found, it is reset to 0.
- **h3** H3 count: Each H3 tag found increments this count by one. However, if a higher level heading (H1 or H2) tag is found, it is reset to 0.
- **h4** H4 count: Each H4 tag found increments this count by one. However, if a higher level heading (H1 to H3) tag is found, it is reset to 0.
- **h5** H5 count: Each H5 tag found increments this count by one. However, if a higher level (H1 to H4) tag is found, it is reset to 0.
- **dan** Paragraph (dan-raku) count: Each paragraph increments this count by one. However, if any heading tag (H1 to H5) is found, it is reset to 0. This column appears only when "Paragraphs" or "Sentences" is selected as the "Unit" under the matrix options.
- **bun** Sentence (bun) count: Each sentence increments this count by one. However, it is reset to 1, when a new paragraph starts, and to 0, if any heading tag (H1 to H5) is found. This column only appears when "Sentences" is selected as the "Unit" under the matrix options.
- **id** Document sequence number: This number is incremented by one for each document found, and is not reset.
- **name** Document heading: Lists any text in the data enclosed in H1 to H5 tags. This column only appears when a heading tag (H1 to H5) is selected as the "Unit" under options. It will not appear, if either "Paragraphs" or "Sentences" is selected as the "Unit" instead.
- **length_c** Length of the document given by the number of characters.
- **length_w** Length of the document given by the number of words

Depending on the "Unit" selected by the user (see section A.2.1), the type of columns (variables) exported will change. In particular, columns (variables) representing units at lower hierarchical levels than the one selected will not be included in the matrix. For example, if "H3" is specified as the "Unit", then "h4", "h5", "dan", and "bun" columns will not be present. Else if "Paragraphs" or "Sentences" is specified as the "Unit", then "name" will not be present.
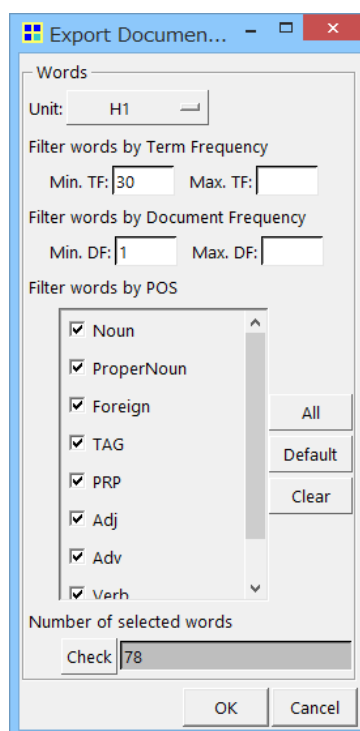
Figure A.43   Export Document–Word Matrix window

Options

■**Units**   In the options under this command (Figure A.43), you need to specify the unit defining a single document, as described in section A.2.1.

■**Filtering Words**   A "document-word" matrix using all words that appear in the document will likely result in a list of several hundred thousands of words, making both reading the data and running multivariable analyses difficult. Hence, we recommend using filtering options to reduce the number of words exported. Typically, words can be filtered using three criteria: TF, DF, and POS (see section A.5.8). Clicking the [Check] button will display the number of words, once filtering has been applied (Figure A.43). Filtering can remove words that seldom appear, as well as words that appear frequently in too many documents. Such words are not usually suitable for statistical analysis and removing them facilitates your analysis.

In the default window, a number with a multiple of 5 automatically appears as the default value for "Min. TF". This sets the number of words to be exported to around 75. When analyzing Japanese text data, check boxes for "名詞 B" (Noun B), "動詞 B"(Verb B), "形容詞 B" (Adjective B), "副詞 B" (Adverb B), "否定助動詞"(Negative Auxiliary Verb), and "形容詞（非自立）" (Adjective (auxiliary)) are initially unchecked. Any default settings can be modified as desired.

■**SPSS Format**   When "SPSS File" is selected as the export format, two SPSS syntax files are created (with extensions *.sps): one with a specified file name, and the second with the file name followed by "_Conv". Running the first syntax file in SPSS creates the same data matrix as the one shown in Figure A.42, while running the second syntax file (Conv) switches the columns and rows, as shown in Figure A.44. The second syntax file operation can be executed, as desired.

Running this command also creates a data file (with extension *.dat) with the specified file name. Since the SPSS syntax file created under this command has the SPSS command for reading information

	CASE_LBL	Word	POS	var001	var002	var003
1	w0	school	Noun	14	20	15
2	w1	room	Noun	6	29	20
3	w2	teacher	Noun	1	17	13
4	w3	house	Noun	23	7	9
5	w4	time	Noun	12	8	13
6	w5	fellow	Noun	4	7	10
7	w6	day	Noun	15	9	12
8	w7	student	Noun	1	3	13
9	w8	way	Noun	8	8	7
10	w9	night	Noun	3	2	5
11	w10	head	Noun	5	3	4

Figure A.44   Document-Word Matrix with columns and rows switched on SPSS

from this data file, deleting this file makes it impossible to execute the syntax file.

■**Variable-Length CSV file for WordMiner**   This format has been added for users that wish to carry out text data analysis with the software "WordMiner". Given that the correspondence analysis in KH Coder generates only a two-dimensional scatter diagram, a more detailed exploration using Word-Miner may be desirable. WordMiner has a rich set of multidimensional data analysis features, including a function for summarizing the characteristics of each component in an easy-to-read table.

It is important to note that KH Coder and WordMiner use different methods for extracting words. When words are extracted differently from the same data, results are difficult to compare. However, using this command enables you to use the words extracted by KH Coder in WordMiner for statistical analysis. This is achieved by exporting the results from KH Coder in a variable-length CSV-format file, which can be read by WordMiner.

In this way, KH coder users can use WordMiner as a statistical analysis tool. Likewise, WordMiner users can employ the advantages of KH Coder for extracting words from their data, since WordMiner analyses do not use POS information, useful for extracting only nouns or only adjectives.

Table A.8   Example of Variable-Length CSV Format

Document ID	Noun	Adj	Verb
Document 1	school teacher fellow	old	say come
Document 2	house face	good bad	say say think
Document 3	school school student	hot	ask

A simplified example of a variable-length CSV format is shown in Table A.8. Although in other formats, each cell shows only the count of a word's appearances, in this format the words themselves are displayed in the cells. All words with a particular POS classification in a given document are listed in a single cell, delimited by spaces. Thus, if the same word appears twice within a document, then this word is written twice in a single cell. Given that each POS is assigned a column (variable), it is easy to perform operations with WordMiner to analyze words from a specific POS.

When WordMiner reads this file, it creates "source variables", such as "noun", "adjective", and "verb". To analyze words in WordMiner, it is necessary to first create "component variables". To perform this operation, select [Create variable] > [Create component variable] > [Change variable type and create new variable]. Since the words have already been extracted by KH Coder, it is not necessary to repeat this step in WordMiner. You only need to change the variable type to create component variables. Likewise, you can select words for analysis using POS. This is carried out with [Create component variable] > [Merge component variable and create new variable] and choosing the desired POS.

It is important to note that in the exported file, conjugated words are converted to their basic forms.

A function for exporting conjugated words in their original form is provided in KH Coder under the menu [Tools] > [Plugin].

A.6.8 [Export Word–Context Matrix]

Overview

This command creates and exports data as a matrix suitable for clustering or mapping words. The data matrix created here also can be obtained by transforming the type of data matrix shown in Figure A.42. However, because such transformation requires programming knowledge, this function has been included to facilitate data analysis. The data format options available are: [CSV File], [SPSS File], and [Tab Delimited].

This function generates vectors representing the textual context of each of n words that are to be clustered or mapped. Let ci be the vector representing the context in which word i is used. The output data is a matrix of n rows with each row consisting of $c1, c2, \ldots ci, \ldots cn$. The vector ci is created using m words, from $word_1$ to $word_m$. Let ej be the average number of appearances of word j in a sentence where word i appears. Then, the "context vector" of word i calculated for each sentence (bun) bi is represented as $(e1, e2, \ldots ej, \ldots em)$. Similarly, di is the "context vector" of word i calculated for each paragraph (dan-raku), while $hi5$ to $hi1$ are the "context vectors" calculated for heading tags H5 to H1. Thus, ci is represented as:

$$\vec{c}_i = \alpha_1 \vec{b}_i + \alpha_2 \vec{d}_i + \alpha_3 \vec{h}_{i5} + \alpha_4 \vec{h}_{i4} + \alpha_5 \vec{h}_{i3} + \alpha_6 \vec{h}_{i2} + \alpha_7 \vec{h}_{i1}$$

Where $\alpha 1$ to $\alpha 7$ are constants that can be freely specified. These output data in the form of an n × m matrix (Figure A.45). Thus, "context vectors" can be used to show what other words are closely associated with a given word.

	A	B	C	D	E	F
1	words(TF)	cw: school	cw: room	cw: teacher	cw: house	cw: time
2	school(130)	424.929	49.263	98.152	55.463	41.570
3	room(119)	60.917	433.272	89.228	77.908	47.971
4	teacher(119)	94.638	83.809	456.088	36.959	30.639
5	house(108)	66.701	77.877	38.348	444.995	45.254
6	time(95)	58.288	60.479	45.985	43.924	393.927
7	fellow(87)	86.848	51.526	76.390	35.662	19.886
8	day(84)	104.053	74.189	65.136	73.189	58.712
9	student(84)	120.454	73.061	141.578	48.617	28.283
10	way(79)	61.195	55.610	28.818	53.610	36.429
11	night(69)	84.833	76.903	70.792	64.660	52.861

Figure A.45   Partial view of a Word–Context Matrix

Although you can perform clustering and plotting using the data in the form shown in Figure A.42 generated without using this command, using context vectors has several advantages. First, clustering and plotting too many words can make the results difficult to read. Therefore, the number of words available for analysis is at most several hundred. However, using context vectors, m words can be used to represent context, in addition to the n words used for clustering and plotting. Even though the number of words used for clustering and plotting is at most several hundred (n words), clustering and plotting can process data on a larger number of words (m words). Hence, it is possible to include information even on low frequency words. In general, the less frequently words appear, the more special or distinctive the words. Therefore, using context vectors has a great advantage in being able to reference the appearance patterns of these distinct words through their context vectors. Second, weighting can easily be added to your analysis. For example, if you are analyzing newspaper articles, and suppose several words are

used in the same article, then the degree of association between those words is considered to be "one". If these words are used in the same paragraph, then the weight is twice this, while if they are used in the same sentence, then it is four times this value. Such weights can be specified by designating values to the constants α 1 to α 7 mentioned above.

Options

■**Filtering words**   In the "Words" area in the upper left of the Export Word-Context Matrix window, you can choose n words for clustering or mapping. Words can be reduced through filtering, using three criteria: TF, DF, and POS (see section A.5.8). Clicking the [Check] button on the bottom left will show you the number of words retained after filtering using the current settings. These operations are equivalent to those for the [Export Document-Word Matrix] command (see section A.6.7).

■**Filtering words for constructing context vectors**   In the "Words for Calculating Context Vectors" area on the right of this window, you can select m words to be used for creating context vectors. Just as above, words can be filtered using "TF", "DF, and "POS" criteria.

■**Units and weights**   In the "Units & Weights" area at the bottom left of this window, you can choose the units for calculating context vectors, and specify weights for the constants α 1 to α 7. The units for calculation must be checked, corresponding to "Sentences", "Paragraphs", "H1", "H2", etc. Multiple units can be selected at one time. Values for the α constants can be entered in their boxes, with "1" set as initial values.

A.7   [Tools] > [Coding] menu

A.7.1   [Frequency] of codes

Overview

This command executes coding according to the contents of the coding rule file (section A.2.5), and provides the following tabulation: (1) the number of documents each code applies to and its percentage of the total, and (2) the number of documents in which no code applied and its percentage of the total. Based on these percentages, the user can determine how completely documents were coded.
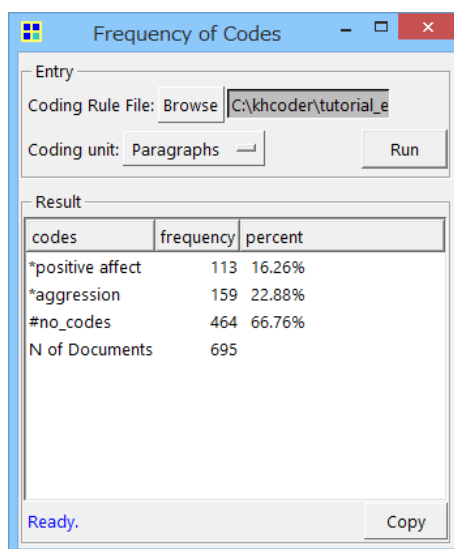

Figure A.46   Frequency of Codes window

Options

Using this command opens the Frequency of Codes window shown in Figure A.46. Clicking the [Browse] button opens a window, where you can specify a coding rule file. In the Windows version of KH Coder, you can also drag a coding rule file and drop it onto the [Browse] button or the file name field. The button just to the right of "Coding unit" specifies the desired coding unit. Finally, clicking [Run] displays the tabulation results.

A.7.2 [Crosstab] of codes

Overview

Using this command, you can code text data according to a coding rule file, divide the data into parts, and perform tabulation on each part. If the target file is a book, the percentage of sentences to which each code applies can be tabulated for each chapter. For HTML-marked newspaper article data, as shown under "A.2.1 Data Preparation", selecting "H3" for "Coding Unit" and "Heading 1" (H1) for "Crosstab" will tabulate the percentage of articles to which the code applies with respect to year of publication.

The menu button to the right of "Crosstab" lists all the variables (section A.8.1) currently available, as well as higher level headings (like H1 to H3). For responses to open-ended questions in questionnaires, you can also tabulate by gender, academic background, or other socioeconomic parameters. If the value of a variable or value label is "MISSING", ".", or "欠損値", then it is treated as a missing value.
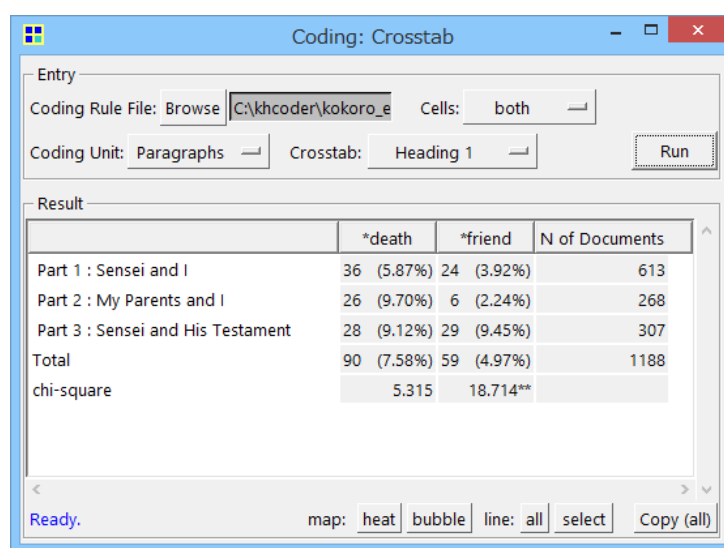

Figure A.47 Crosstab of Codes window

If the ratio of documents with a specific code applied changes significantly, then one or more asterisks will appear after their chi-square values. Two asterisks indicate an alpha level of 0.01 (i.e., p < 0.01), while one asterisk indicates an alpha level of 0.05. Figure A.47 shows cross-tabulating data with categories "Part 1", "Part 2", and "Part 3". In this example, there have been significant changes in the frequency ratio of the "friend" code at an alpha level of 0.01. For tabulation of a variable with only two categories, like "Part 1" and "Part 2", the Yates' continuity correction (Yates 1934) is automatically applied to the calculation of chi-square.

Options

Executing this command opens the Coding: Crosstab window, like the one shown in Figure A.47. The procedures for specifying the "Coding Rule File" and "Coding Unit" is the same as for the [Frequency] command, described in section A.7.1. This function also requires a cross-tabulation target, which is set by clicking the menu to the right of "Crosstab". Clicking the menu to the right of "Cells" allows you to choose how the tabulation is displayed: [frequency] to show the number (frequency) of documents to which each code applies; [percentage] to show the ratio (percentage) of documents to which each code applies; or [both] to display both values.

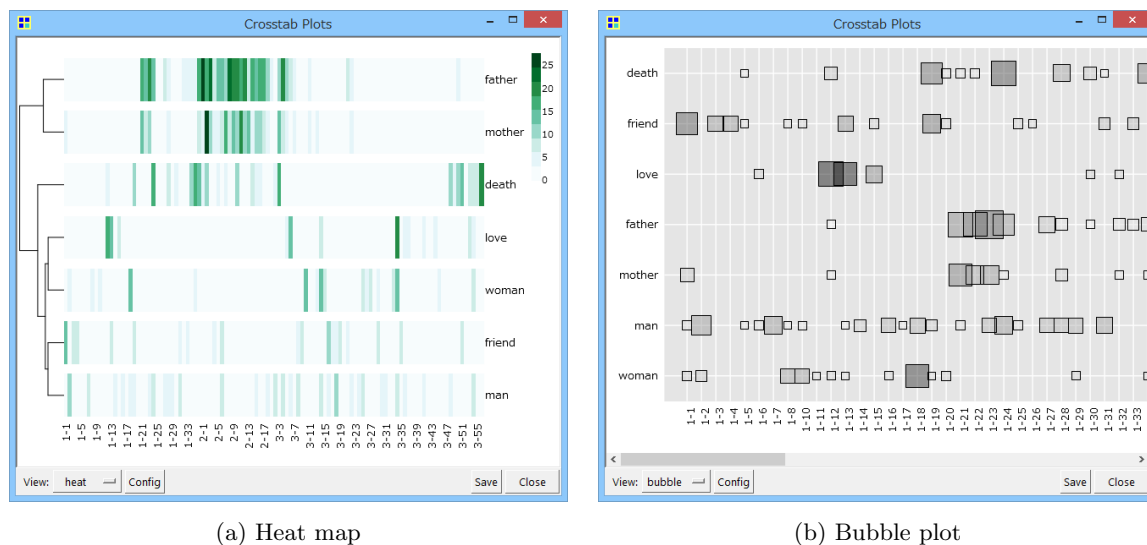(a) Heat map   (b) Bubble plot

Figure A.48   Cross-tabulation charts in KH Coder

This command also allows you to display the tabulation results as a heat map or bubble plot (shown in A.48(a) and (b), respectively), or a line chart. On a heat map, frequency is indicated using a gradient color scale, hence, wherever a code appears frequently has a darker color. This type of information can also be shown with hierarchical cluster analysis that applies the Ward method and uses the Euclidian distance for clustering, as shown in Figure A.48(a). One important feature of heat maps is that they provide an overview of the entire text file on a single screen. However, if the chart includes a large number of codes, then vertical scrolling may be required.

On a bubble plot, the sizes of squares or circles are used to represent the ratio of documents to which each code applies. The different-sized squares or circles on a bubble plot provide better visibility, than the varying color shades on a heat map. However, for large numbers of documents, you would need to scroll the screen horizontally to see all data.

In the Crosstab Plots window, displaying a heat map or bubble plot, the options shown on the Configure: Crosstab Plots window (Figure A.49) can be accessed by clicking the [Config] button on the bottom left. Here, you can select which codes are displayed in the chart, whether to show the results of your cluster analysis as a heat map, and much more. For the bubble plot, you can select whether to display bubbles as circles or squares, and whether to visualize the standardized residuals with colors or a grayscale. Because the plot can be saved in several formats, it is compatible with various types of software (see section A.5.4, subsection "Saving and using plots").
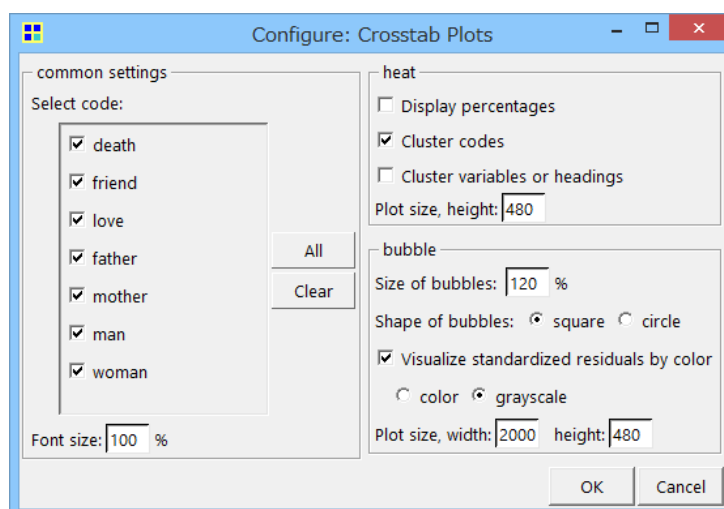
Figure A.49 Options for cross-tabulation charts

A.7.3 [Similarity Matrix] of codes

Overview

This command performs coding according to the coding rule file and calculates the degree of association between codes. To determine their associations, the value of their Jaccard similarity measure is calculated and displayed. This value varies between 0 and 1. Codes that appear frequently in the same document are estimated to be closely associated, and their values approach unity.
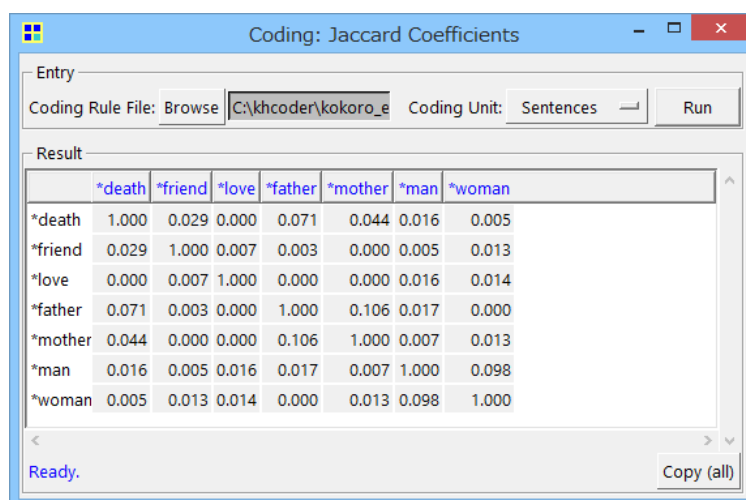
Figure A.50 Similarity matrix of Jaccard coefficients

Options

Executing this command opens a window like the one shown in Figure A.50. The operation and options for this window are exactly the same as those for the [Frequency] command (see section A.7.1). In Figure A.50, clicking a column code name sorts the matrix in descending order based on the values in the selected column. Re-clicking the same code name restores the original similarity matrix.

A.7.4   [Correspondence Analysis] of codes

Overview

This command performs a correspondence analysis on codes and presents results in a two-dimensional scatter diagram. This command is used to explore which kinds of codes have a similar appearance pattern; which codes are considered characteristic for each value of a variable, or for each chapter or text section; and which chapters or values of variables are similar, based on the codes that are defined for these text data.
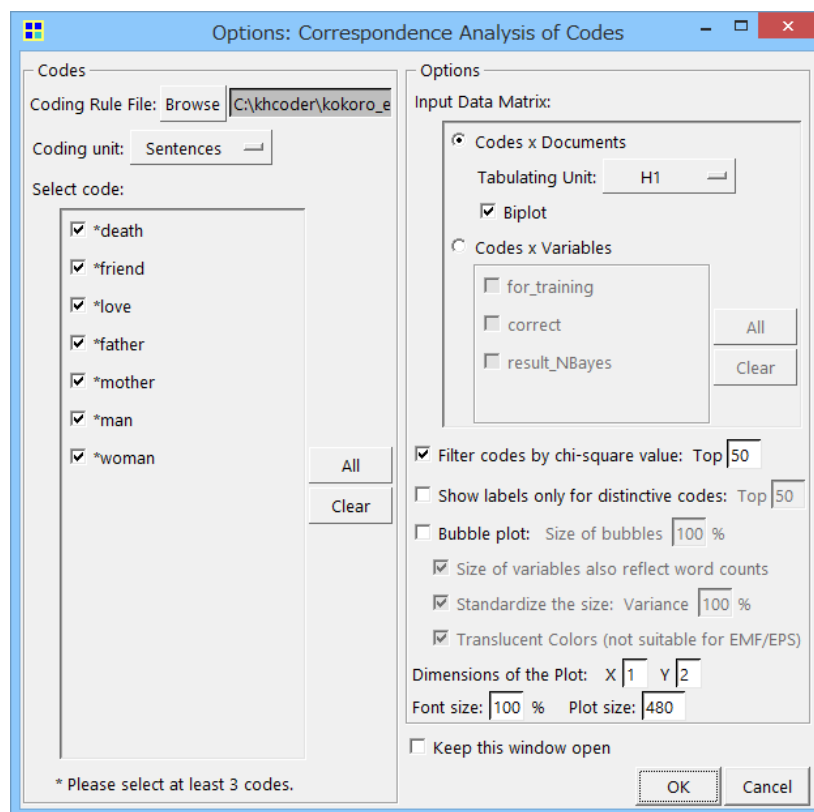

Figure A.51   Correspondence analysis using coding results

Options

■**Selecting codes**   In the Options window shown in Figure A.51, you first need to select a coding unit and specify a coding rule file on the left side. The specific codes you wish to use for analysis can then be selected using the checkboxes next to each code listed in the "Select code" area. A minimum of three codes is required to perform this analysis.

■**Data matrix type**   The starting point for a correspondence analysis is the data matrix, which can be generated to reflect different analysis goals. To search for codes with a similar occurrence pattern, or to explore what kind of components can be found in the appearance pattern of each code, we recommend you select "Codes × Documents" on the right side and choose a tabulating unit that matches the coding unit on the left. This analysis will use the matrix generated with the [Export Document-Code Matrix] command (see section A.7.9), with positions and document lengths removed.

To investigate the differences between chapters and sections, select "Codes × Documents" and choose a tabulating unit that is at a higher level than was used for the coding unit. To investigate the differences

between the values of a variable, select "Codes × Variables" and then select the desired variable. All these analyses use the matrix created by the [Crosstab] command (see section A.7.2). As for cross-tabulation, if a value or value label is "MISSING", ".", or "欠損値", then it is treated as missing.

■**Common Operations** Other operations and options similar to those available for the [Correspondence Analysis] under Words command (see section A.5.8) are also available here. Thus, you can specify how many components to extract, and which components to assign to the x- and y-axes, as well as change font size, plot size, and other plot variables. Simply use the [Config] button at the bottom of window, once your analysis has been performed. Scatter diagrams can be saved by clicking the [Save] button on the bottom right (see section A.5.4, subsection "Saving and using plots").

A.7.5   [Multi-Dimensional Scaling] of codes

This command enables you to draw one- to three-dimensional scatter diagrams to explore associations between codes, rather than using a similarity matrix. The graphical representation is often easier to interpret than data in a similarity matrix. The operation and options here are similar to those for the [Multi-Dimensional Scaling] under Words command (see section A.5.9). The only difference is that codes rather than words are used for analysis. A minimum of five codes is required to perform this command.

A.7.6   [Hierarchical Cluster Analysis] of codes

This command enables you to use a dendrogram to explore code associations. In some cases, a dendrogram may be easier to interpret than other analysis outputs. The operation and options for this command are similar to those for the [Hierarchical Cluster Analysis] under Words command (see section A.5.10). The only difference here is that codes rather than words are used for analysis. A minimum of three codes is required to perform this command.

A.7.7   [Co-Occurrence Network] of codes

This command enables you plot a network diagram (graph) to explore code association. Given that codes with similar appearance patterns are directly connected with lines in a network, it may be easier to interpret these visual results, compared with other analyses. The operation and options here are similar to those for the [Co-Occurrence Network] under Words command (see section A.5.11). The only difference is that codes rather than words are used for analysis. A minimum of three codes is required to perform this command.

However, care should be taken when setting the parameter for filtering edges. The default number of edges is set to "Top 60", which is suitable for analyzing more than approximately 60 to 70 words. This is too large a value for processing, for example, 5 to 10 codes. When using 5 to 10 codes for analysis, set this value between "Top 5" and "Top 10", or else filter data using a Jaccard coefficient, by selecting "Jaccard coef. >=" and entering a number like 0.1 or 0.2 in the box.

A.7.8   [Self-Organizing Map] of codes

This command enables you to use a self-organizing map to explore code associations. The operation and options for this command are similar to those for the [Self-Organizing Map] under Words command (see section A.5.12). The only difference here is that codes rather than words are used for analysis. A minimum of three codes is required to perform this command.

A.7.9   [Export Document–Code Matrix]

Overview

This command enables you to export data describing whether each code has been applied to a given sentence, paragraph, section, or document. The data can be exported as a [CSV File], [SPSS File], [Tab Delimited] file, or [Variable-length CSV: for WordMiner] format file.

   The exported file consists of rows (or cases) that list tabulating units, such as a sentence or paragraph, and columns (or variables) that list codes and positions (see Figure A.52). The positions exported here represent the same data that is output from the [Export Document-Word Matrix] command (see section A.6.7). Therefore, the file exported here for codes is similar to that generated by the similar command for words described under section A.6.7. The main difference is that for words, data cells show the number of appearances of words in the unit; while for codes, the data cells show a binary value of 1 or 0 that indicates whether the code is applied or not.

	A	B	C	D	E	F	G	H	I	J	K
1	h1	h2	h3	h4	h5	dan	*death	*friend	*love	*father	*mother
2	1	1	0	0	0	1	0	0	0	0	0
3	1	1	0	0	0	2	0	1	0	0	1
4	1	1	0	0	0	3	0	1	0	0	0
5	1	1	0	0	0	4	0	0	0	0	0
6	1	1	0	0	0	5	0	0	0	0	0
7	1	1	0	0	0	6	0	0	0	0	0
8	1	2	0	0	0	1	0	0	0	0	0
9	1	2	0	0	0	2	0	0	0	0	0
10	1	2	0	0	0	3	0	0	0	0	0

Figure A.52   Partial view of an exported Document-Code Matrix

Options

To execute this command, you only need to specify a coding rule file and a coding unit in the Options window. To reach this window, first choose one of the available export formats under [Tools] > [Coding] > [Export Document–Code Matrix]. If an [SPSS File] is selected, then an SPSS syntax file (with file extension *.sps) with a specified file name is created. You need to run the syntax file in SPSS to load these data and analyze them with SPSS. For further details on the variable-length CSV format, see section A.6.7 ([Export Document-Word Matrix]).

A.8   [Tools] > [Variables and Headings] menu

A.8.1   [Import Variables]

Overview

Importing additional attributes of the target text as "variables" allows you to use it in commands, such as [Word Association], [Corresponding Analysis], [Co-occurrence Network], and [Search Documents], as well as in coding rules. For example, in the case of newspaper articles, publication years or page numbers can be imported as variables. This allows you to analyze how the contents of newspaper articles change, depending on publication year and page number (using commands like [Word Association], and [Corresponding Analysis]), and to attach a code only to articles on a specific page. Likewise, if data are responses to an open-ended question in a questionnaire, then using variables, such as gender or age, allows you to perform a more relevant analysis.

   Calling this command opens the window shown in Figure A.53. Click the [Browse] button, and select a file that contains variables. Select the "Unit of Variables" used to import this variable, and then click
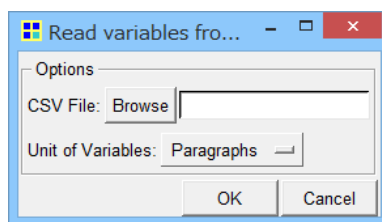
Figure A.53   Read variables interface window

[OK] to continue. In the example of the newspaper articles mentioned in section A.2.1, with a variable file containing the page numbers of each article, you should select article unit (i.e., "H3"). Either a CSV file or a tab-delimited file may be imported.

Files of variables must be in either CSV or tab-delimited format, and a variable name should be recorded in the first row. In addition, the number of cases (the number of documents) that KH coder recognizes in these data and the number of cases (the number of lines) in the file must match to import variables. The type of variables can be numeric or string; however, neither variable names and values cannot exceed 255 characters. Given that all of the full-width spaces included in the file will be converted to the half-width spaces, full-width versus half-width spaces are not distinguished. There are a number of restrictions for variable names: they cannot start with an underscore (_), have half-width quotation mark(s) (""), or use "heading 1" to "heading 5". For the preparation of a file containing variables, please refer to tips provided on the web page.

Internal Processing

A table that contains the imported variable name, unit, and the name of the table that contains the actual values is saved as an "outvar" table in the MySQL database. To view actual values, tables with names like "outvar0" and "outvar1" will be created and saved in the MySQL database.

A.8.2   [List] of variables

Overview

Calling this command opens a List of Variables & Headings window like the one shown in Figure A.54. This window allows you to check variables that are already imported, as well as the unit selected when each variable was imported. Using [Delete] or [Export], any unnecessary variable can be deleted, or the contents of a specific variable can be exported to a CSV file. Since KH Coder saves the results of your cluster analysis (Section A.6.2) and Naive Bayes classifier (Section A.6.3) as variables; you can use the [Export] button to retrieve such results.

KH Coder recognizes headings defined with H1 to H5 tags as a special type of variable. Therefore, headings of the form "Heading 1" to "Heading 5" occur in the same list, as variables. Headings can be used in much the same way as variables; however, they differ from variables in that they cannot be deleted or labeled.

When you click on a specific variable in the left pane of the List of Variables & Headings window (Figure A.54), the details on this variable will be displayed in the right pane. Here, it will show variable values, frequencies (i.e., the number of documents (cases) containing the value), and value labels. Value labels also can be added or edited in this window.

In addition, by clicking on the [Documents] button, after selecting a value, will open the Search Documents window, where a list of the documents with the selected value will be displayed. Similarly, clicking on "*Words" and then selecting a "selected value," opens the Word Association window (Section A.5.7), displaying a list of higher-frequency words in documents with the selected value, i.e., distinctive
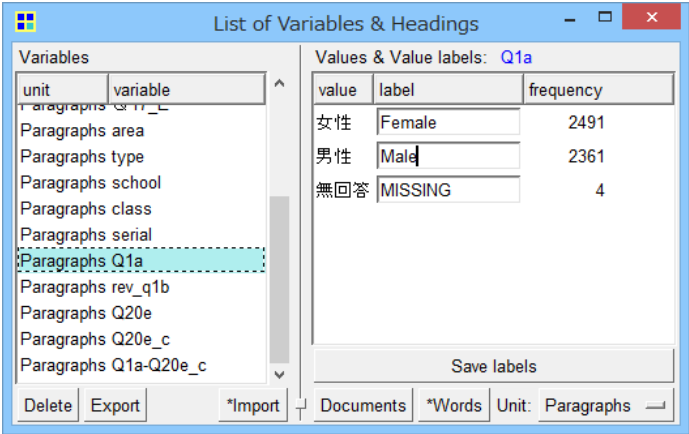
Figure A.54   Variables, values & value labels

words for the selected value. These two operations can also be executed by double-clicking on the value, and double-clicking on the value while holding down the shift key, respectively.

Figure A.55   Distinctive words for each value

Furthermore, a table (Figure A.55) listing the distinctive words for each value can be created by clicking the [*Words] button, and selecting "catalogue: Excel". In the Figure A.55, data from responses to open-ended questions are compared by gender, and the distinctive words for each value (i.e., gender) are listed in two separate panels. In this menu, there is a choice between "catalogue: Excel" and "catalogue: CSV." The appearance and format of the table is similar to that shown in Figure A.55, if you select Excel, but not if you select CSV. To automatically create this type of table, the [Word Association] function (Section A.5.7) is used to search for distinctive words. The settings applied to the [Word Association] function are those used when you click on the [catalogue] button. If you want to change, for example, the selection of a POS, then you will need to set up a filter using the Word Association window, before clicking the [catalogue] button.

Internal processing
Labels attached to variable values are saved in an "outvar_lab" table in the MySQL database. When you click on the [*Words] button, and then select "catalogue: Excel," a temporary file named "coder_data/*_temp0.xls" (*: name of the file targeted for analysis) will be created in this location as the target file. Instead, selecting "catalogue: CSV," creates the filename extension *.CSV. If a file with the same name already exists, a 'temp0' part is replaced with incremental numbers for each new version of the file. All of these files will be deleted, when the same project is opened again.

A.9　[Tools] > [Convert the Target File] menu

A.9.1　[Extract Partial Text]

Overview

When analyzing text data, it is sometimes necessary to focus on a certain part of the data, e.g., only on headings, or on the documents or articles that satisfy particular conditions. Hence, this command enables you to extract documents that satisfy particular conditions and to save them to a new file. By registering this file in KH Coder as a new project, you can analyze a subset of the original data.

Invoking this command opens the window shown in Figure A.56. Its options enable you to extract only the headings contained in a target file or documents that meet particular conditions.
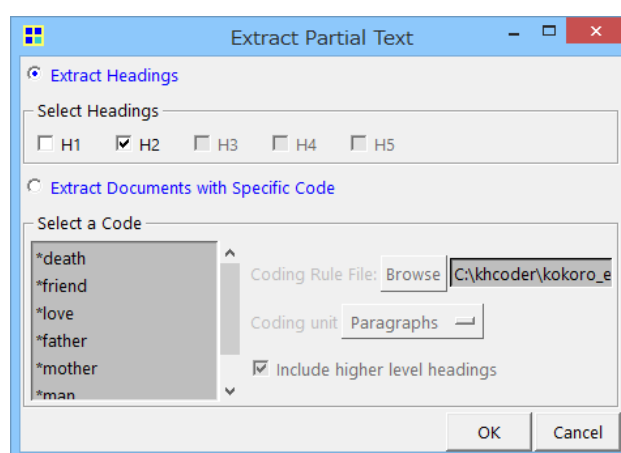

Figure A.56　Extract Partial Text window

■**[Extract Headings]**　To retrieve only headings from a target file, click the [Extract Headings] radio button and custom select the heading levels to be extracted. Using the example of the newspaper article described under section A.2.1 "Data Preparation", selecting only "H3" lists all article headings.

■**[Extract Documents]**　To extract documents that satisfy certain conditions, it is necessary to create a coding rule in advance that gives a code to all the documents you wish to extract. The procedure for coding and extracting these documents is: (1) click the [Extract Documents with Specific Code] radio button, (2) specify the coding rule file, and (3) select both a code to be designated as the condition and a coding unit. Following these steps, only documents that have the selected codes will be extracted to a text file.

By selecting the [Include higher level headings] box you can determine whether to include higher-level headings in the new text file. For example, when extracting documents with H3 headings, selecting this check box would also list the H1 and H2 headings.

A.9.2　[Convert to CSV]

If variables have been loaded into the project, they are also partially extracted. These extraction results are saved as a CSV file named "_var.csv". This file can be used to import variables when registering the extracted documents in KH Coder as a new project.

The data format used in KH Coder with tags from H1 to H5 has the advantage of being able to handle the hierarchical structure of text data. However, not many software packages support this data format.

This command converts a target file into CSV format by wrapping a specified unit of data in each row (case). These converted data can easily be imported into commercial text data analysis software and text mining software.

	A	B	C	D
1	2004	Jan	The heading of article 1	The body of article 1 ....
2	2004	Jan	The heading of article 2	The body of article 2.......
3	2004	Feb	The heading of article 3	The body of article 3.......

Figure A.57   Sample conversion to CSV format

Figure A.57 indicates the results of converting the newspaper article data with HTML markings shown in section A.2.1 into CSV format, such that each row consists of a unit of an H3 heading (individual article). In this example, column A lists the H1 heading (year), column B lists the H2 heading (month), column C lists the H3 heading (article heading), while column D shows the article text, with return characters removed.

A.10   [Tools] > [Plugin] menu

A.10.1   [Sample] plugins

The source code of KH Coder is open to everyone. You are able to edit the source code to modify existing functions and add new functions, as required. However, reading and understanding the entire source code to modifying it requires much effort. To circumvent this, a plug-in mechanism is provided. Using this mechanism, you can add new commands to KH Coder by simply creating relatively short Perl scripts.

The subfolder "plugin_en" is located in the same folder as the executable file "kh_coder.exe". All Perl script files created in this subfolder are recognized by KH Coder as plug-ins. Note that, if the user interface language is set to Japanese, then KH Coder reads such plug-ins from the "plugin_jp" folder.

All custom-written Perl scripts to be used as plug-ins by KH Coder must have the file extension ".pm" and be located in a Perl module format that contains the Perl package. In addition, each script requires two subroutines: the "plugin_config" to add a command to the KH Coder menu, and the "exec" to execute the command when it is selected.

A number of examples are provided in this subfolder. For example, "sample1_hello_world.pm" is a simple script that displays "Hello World!" on the screen. The "sample2_exec_sql.pm" script shows the user how to use the MySQL database, while the "sample3_exec_r.pm" script shows a method for passing commands to R to run them. Simply editing these simple scripts, will allow you to build your own customized commands to perform a customized search and automatically save these results to a text file. However, some knowledge of the Perl programming language is required to write such scripts, as well as knowledge of SQL statements to use a MySQL database. For Perl and SQL programming support, refer to their respective reference documents, while to learn more about the format of the MySQL database created by KH Coder, see section A.4.2 [Run Pre-Processing].

A.10.2   [Unify *.txt Files in the Folder]

The functions in KH Coder are designed under the assumption that all text data to be analyzed will be contained in a single file. Hence, in a case where the target data is saved as 20 separate files, it is necessary to combine these files into one file to analyze the contents with KH Coder. This command (Figure A.58) automates this task.

All (and only) text files (*.txt) located in the folder specified in the Options window in Figure A.58
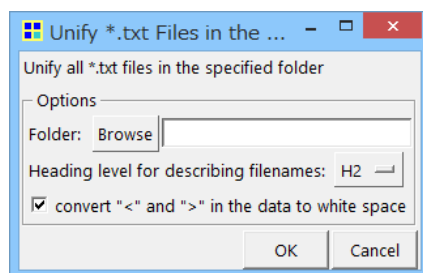
Figure A.58   Unify *.txt Files in the Specified Folder window

will be unified into a single file. A heading in the format "file: " for each text file read is inserted as a delimiter. Using this heading, even after all files are unified into one file, each original file can be treated as a separate "document" in KH Coder. The heading level can be changed from "H1" to "H5". If no file contains "H1" to "H5" headings, we recommend checking the option 'convert "<" and ">" in the data to white space' .

Because headings are added to delimit the data, the numbers of counted characters and words will be different from sum of the original files. To account for this, another file called "_names.txt" is created each time this command is executed. This file lists the headings in the format "file: " inserted by executing this command in separate lines. You can drag and drop this file into the input area of the "force pick up" field under the [Select Words to Analyze] command (see section A.4.3) and its contents will appear in this field. Under this setting, all headings created by this command are extracted as a single word. If the contents are also pasted in the "force ignore" field, then headings are not counted and will be treated as nonexistent in any statistical analyses.

A.10.3   [New Project with Blank Line Support]

Currently, this plugin command is only available in the Japanese user interface.

Overview
Similar to the [New] command under the [Project] menu (see section A.3.1), this command is for registering a new target file in KH Coder. While the blank lines in a target text file are generally treated as nonexistent, this command recognizes each blank line as a paragraph (case).

For data dealing with open-ended questions in questionnaires, if a blank line indicates no response or a missing response, then this command can be convenient. If no response or missing responses are treated as nonexistent, then the number of cases decreases. When this happens, you will have trouble reading variables, or combining data exported from KH Coder with the responses to other questions.

Since blank cells in Excel files are recognized in default, this plugin command is only useful for text format files.

Internal processing
Executing this command creates a new text file in the folder that contains the target file specified by the user. The new file is a copy of the specified target file, but with its empty lines replaced with the text string "---MISSING---". This new text file can be registered with KH Coder as a new target file. The text string is configured to be extracted forcibly as a single word, and then ignored using the [Select Words to Analyze] command (see section A.4.3). The name of the newly created and registered text file can be viewed in the main window of KH Coder (Figure A.4).

A.10.4   [Export Document-Word Matrix (Surface Form): Variable-length CSV (WordMiner)]

Currently, this plugin command is only available in the Japanese user interface.

Executing this command achieves almost the same result as the [Variable-length CSV: for WordMiner] command under [Export Document-Word Matrix] (see section A.6.7). However, this command will output the extracted words as they are, even if conjugated or inflected forms are present; it does not convert them to their base forms.

A.10.5   [Reload Morphological Analysis Results]

To customize how words are extracted in KH Coder, it may be sufficient to specify strings to be extracted forcibly as single words using the [force pick up] field (see section A.4.3) or to edit the dictionaries used in ChaSen, MeCab, or other POS taggers. However, you may prefer changing the word extraction results more flexibly. In such cases, you may choose to modify the result file manually and to execute this command. The result file is stored in the "coder_data" folder, with a name consisting of the target file name followed by "_ch.txt".

Executing this command achieves almost the same results as [Run Pre-Processing] (see section A.4.2). The only difference is that this command does not perform morphological analysis with ChaSen, MeCab or the POS tagger. The command exports a manually modified morphological analysis results to a MySQL database and prepares the data. Therefore, all the subsequent processes (searching, coding, statistical analysis, etc.) are carried out on the modified data.

We strongly recommend creating backup files frequently when you modify word extraction results, because the original result file (coder_data\[target file name]_ch.txt) is overwritten without warning, whenever the [Run Pre-Processing] command is executed. If this happens before creating a backup file, manually modified data will be lost, when it is overwritten by the latest results of a morphological analysis performed by ChaSen, MeCab or the POS tagger. Therefore, we strongly advise the user to create a backup file by copying the original result file, immediately after making any modifications manually.

A.10.6   [Word Clusters: UniDic]

This plugin command is only available in the Japanese user interface. It is useful only when target language is Japanese.

The POS system used in UniDic is different from that of IPADIC. Therefore, if the dictionary of ChaSen or MeCab has been changed from the standard IPADIC to UniDic, then the [Word Clusters] commands (see sections A.4.4 and A.4.5) under [Pre-Processing] will not be functional. In this case, you can use this command to extract word clusters.

A.11   [Tools] menu

A.11.1   [Execute SQL Statements]

Overview

KH Coder stores all the results of your morphological analyses, such as word extraction by ChaSen, in MySQL databases, and performs all actual data retrieval, and coding using MySQL. Therefore, sending SQL statements (commands) directly to MySQL can facilitate various types of searching and tabulation processes outside the scope of KH Coder. This command was developed to act as a simple interface between KH Coder and MySQL to allow the user to send SQL statements to MySQL, enabling more

flexible data retrieval and tabulation. For more technical options within MySQL, we advise you to use the dedicated client software, with more extensive functions. For a description of the table structure of MySQL databases, please refer to section A.4.2.

Although KH Coder has a [Search Words] command (section A.5.5), this does not allow the user to narrow data using their POS names, e.g., selecting only verbs as the search target. However, as shown in Figure A.59, executing a series of SQL statements via this command achieves such a search. Here, words containing "th" are searched for among extracted verbs and displayed in descending order of frequency.
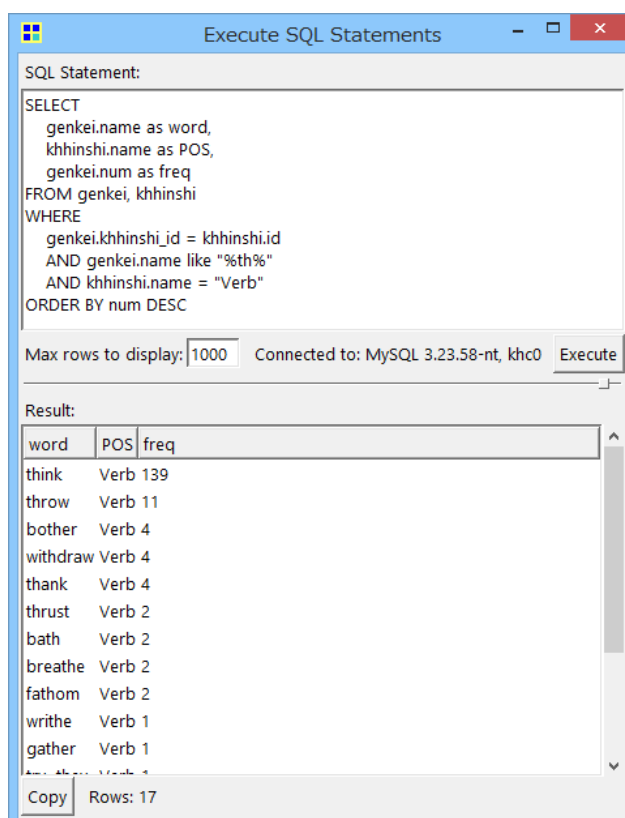

Figure A.59   Example of an execution of the SELECT statement

Option

To execute more than one SQL statement, add a semicolon (";") to the end of each statement and carry out two newline operations immediately after the semicolon, before entering the next statement. Given thousands or even tens of thousands of records retrieved, KH Coder is configured to display only the first 1,000 records. To change this setting, enter a new value into the [Max rows to display:] textbox, shown in Figure A.59.

A.12   [Help] menu

A.12.1   [Manual (PDF)]

Executing this command opens a PDF reference manual. To view this file, a PDF reader must be installed beforehand. The document you are now reading is the English version of the very same manual.

A.12.2   [Latest Info (Web)]

Executing this command navigates the user to the official website for KH Coder. To browse this website, an Internet connection must be established. The website provides the latest version of KH Coder, support information, and a list of research publications that have used KH Coder for their analyses.

A.12.3   [About]

Executing this command displays a message box with version and copyright information on KH Coder, as well as its logo. The logo design represents the author's hope that: "Even if you are lost in a flood of information, KH Coder will surely help you see through the surface of the water and find your key concepts."

A.13   Miscellaneous

A.13.1   Conditions of use

Although the copyright for KH Coder belongs to Koichi Higughi (the author), this program is free software. KH Coder can be used, modified, and redistributed under the terms of Ver. 2 or later versions of the GNU General Public License (GPL). You can find details of the GPL at Part B of this document.
   Conditions of use are summarized below.

No Guarantee
The author believes that KH Coder is a safe and useful software package. However, this is not guaranteed. The author is exempt from any direct or indirect damage and loss that may incur as a result of the use of KH Coder.

Freedom of use and modification
KH Coder can be used freely for academic, commercial, or other purposes. Customization of the existing functions and addition of new functions by modifying the source code also are permitted.

Freedom of redistribution
KH Coder can be redistributed freely, as long as the following conditions are fulfilled. First, it must be redistributed together with its source code. Second, if it is redistributed, after being modified, then all modifications of the source code must be revealed. Finally, the above freedom to use and to redistribute must be available to any redistributes, while explicitly stating that KH Coder is not under any guarantee.

When KH Coder is used for academic purposes
In addition to its uses prescribed in the GPL, if you publish papers or articles based on research carried out using KH Coder, then we would be grateful if you acknowledge your use of KH Coder in those publications. We would also be very grateful if you could provide the author with bibliographic information on these publications, so as to update publications on our website resources.

A.13.2   Support

In future, the latest versions of KH Coder will be distributed via the following website:

   http://khc.sourceforge.net/en

This website provides the most up-to-date version of KH Coder, as well as support information and a list of relevant publications. Any technical inquiries about KH Coder, bug reports, and improvement requests can be submitted via the user forum located on this site. The forum also has a very useful Q&A section, as a reference for other users wishing to customize KH Coder. Questions posted via this forum have sometimes led to the discovery and correction of bugs, thus providing an important interface for feedback and discussion between the developers and users of this package. Please feel free to post your questions and feedback here.

A.13.3   Source code

The source code for KH Coder is written in Perl. Its Windows-version .exe file created using Perl Dev Kit from ActiveState contains Perl itself and a series of Perl scripts. The .exe file can be executed without installing Perl. However, it is necessary to install this language, if you intend to modify the source code to customize it.

A.13.4   Uninstalling KH Coder

KH Coder does not use the Windows Registry. Therefore, if you intend to uninstall KH Coder, then you only need to delete the files including kh_coder.exe. The fastest way to do this is to delete the top-level folder that was created when you set up KH Coder. The same method can be applied to Linux and Mac versions.

Part B

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free

use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

    Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

    You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

    (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

    (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

    (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

    These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

    Thus, it is not the intent of this section to claim rights or contest your rights to work written

entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other

reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY

SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> one line to give the program's name and a brief idea of what it does.
> Copyright (C) yyyy name of author
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program

'Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Bibliography

Afifi, A. & V. Clark, 1996, *Computer-Aided Multivariate Analysis 3rd ed.,* Boca Raton, FL: Chapman & Hall/CRC.

Clauset, A., M. E. J. Newman & C. Moore, 2004, "Finding Community Structure in Very Large Networks," *Physical Review E,* 70(6): 066111.

Danowski, J. A., 1993, "Network Analysis of Message Content," W. D. Richards Jr. & G. A. Barnett eds., *Progress in communication sciences IV,* Norwood, NJ: Ablex, 197–221.

Fruchterman, T. M. J. & E. M. Reingold, 1991, "Graph Drawing by Force-directed Placement," *Software - Practice and Experience,* 21(11): 1129–64.

Greenacre, M., 2007, *Correspondence Analysis in Practice, Second Ed.,* Boca Raton, FL: Chapman & Hall/CRC.

Jackson, K. M. & W. M. K. Trochim, 2002, "Concept Mapping as an Alternative Approach for the Analysis of Open-ended Survey Responses," *Organizational Research Methods,* 5(4): 307–36.

Kamada, T. & S. Kawai, 1988, "An Algorithm for Drawing General Undirected Graphs," *Information Processing Letters,* 31: 7–15.

Kaufman, L. & P. J. Rousseeuw, 1990, *Finding Groups in Data: An Introduction to Cluster Analysis,* New York: Wiley.

Kohonen, T., 2001, *Self-Organizing Maps 3rd ed.,* New York: Springer.

Newman, M. & M. Girvan, 2004, "Finding and Evaluating Community Structure in Networks," *Physical Review E,* 69: 026113.

岡太彬訓, 2002, 「社会学におけるクラスター分析と MDS の応用」『理論と方法』17(2): 167–81.

Osgood, C. E., 1959, "The Representational Model and Relevant Research Methods," I. d. S. Pool ed., *Trends in Content Analysis,* Urbana, IL: University of Illinois Press, 33–88.

Pons, P. & M. Latapy, 2005, "Computing Communities in Large Networks Using Random Walks," *ArXiv Physics e-prints*, http://arxiv.org/abs/physics/0512106.

Romesburg, H. C., 1984, *Cluster Analysis for Researchers,* Belmont, CA: Lifetime Learning.（＝ 1992, 西田英郎・佐藤嗣二訳『実例クラスター分析』内田老鶴圃.）

Scott, M., 2001, "Comparing Corpora and Identifying Key Words, Collocations, and Frequency Distributions through the WordSmith Tools Suite of Computer Programs," M. Ghadessy, A. Henry & R. L. Roseberry eds., *Small Corpus Studies and ELT: Theory and Practice,* Amsterdam: Benjamins, 47–67.

Trochim, W. M. K., 1989, "An Introduction to Concept Mapping for Planning and Evaluation," *Evaluation and Program Planning,* 12: 1–16.

Yates, F., 1934, "Contingency Table Involving Small Numbers and the χ^2 Test," *Supplement to the Journal of the Royal Statistical Society,* 1(2): 217–35.