CALIBERcodelists user guide

Anoop Shah, Clinical Epidemiology Group

May 17, 2013

User guide for Version 0.2-0.

Contents

1 Introduction

The `CALIBERcodelists` package is designed to handle lists of ICD-10, Read and OPCS codes to define medical conditions for studies using CALIBER or other databases of UK electronic health records. The package is written in R, a statistical analysis and programming language, but many of the functions are available via an interactive menu and do not require experience in using R.

This user guide is intended both within and outside CALIBER. The `CALIBERcodelists` package is licensed under the GNU General Public License Version 3 (`http://www.gnu.org/licenses/gpl-3.0.html`), but CALIBER cannot distribute the Read, ICD-10 and OPCS source dictionaries because of licensing issues. Sections of this user guide relating to conversion of source dictionaries are specifically for non-CALIBER users and can be ignored by people who have access to the `CALIBERlookups` package.

Commands to be typed into the console are typeset in blue typewriter font `like this`, and R output shown in the console is typeset `like this`.

2 What can the package do?

- Provide a standard method for identifying Read, ICD-10 and OPCS codes of interest by searching on term text or codes
- Allow codelists to be viewed in a spreadsheet and individual terms deleted or their categories changed
- Import codelists in a range of formats, and export them in a standard file format
- Compare one codelist with another, or merge two codelists
- Convert codelists from one dictionary to another, using the NHS mapping between Read/OPCS and Read/ICD-10 terminologies (note that this mapping is not considered robust enough for research use without further manual review of the codes)
- Process a document containing code to generate the codelist and descriptive text, generating a detailed HTML document and a codelist in the standardised format

3 Getting help

You can open the CALIBERcodelists help either by searching for 'CALIBERcodelists' in the help window or typing in the console:

```
? CALIBERcodelists
```

You can find help on any function using `?`; for example to obtain help on the `termhas` function, type:

```
? termhas
```

4 Basic R syntax

To execute a command (function) in R, type the name of the function followed by its arguments in brackets. If there are no arguments, you must put an empty pair of brackets. R is case sensitive, and unlike Stata you are not allowed to abbreviate the names of functions.

For example, to call the 'codematch' function to select Read, OPCS or ICD-10 codes, type

```
codematch("I21", dictionary="icd10")
```

or `codematch("I21", "icd10")` – it is optional to include the name of the argument if you are supplying all arguments in order. Single quotes (') can be used instead of double quotes.

Anything on a line after `#` is treated as a comment. If you don't finish an expression (e.g. leave out a closing bracket, or end with an operator such as `+`, `-`, `=`), R assumes the expression continues onto the next line. Text strings such as filenames must be between single or double quotation marks.

Use the `<-` assignment operator to create a named object to stores the output of a function. In this package common outputs are 'codelist' objects and 'selection' objects.

For example, the following code creates a new selection object of terms containing the text 'myocardial infarction' and gives it the name 'mi':

```
mi <- termhas("myocardial infarction")
```

You can then view the selection simply by typing `mi`. When you type the name of an object on its own, R prints it to the console.

If you subsequently assign something else to the name 'mi', the original object will be over-written. Avoid using the names of existing R functions when naming your objects, otherwise you might find it difficult to call the relevant function if you need it (for example, do not create a codelist named 'codelist', use something like 'mycodelist' instead).

Further information on R programming is given in Section 11.

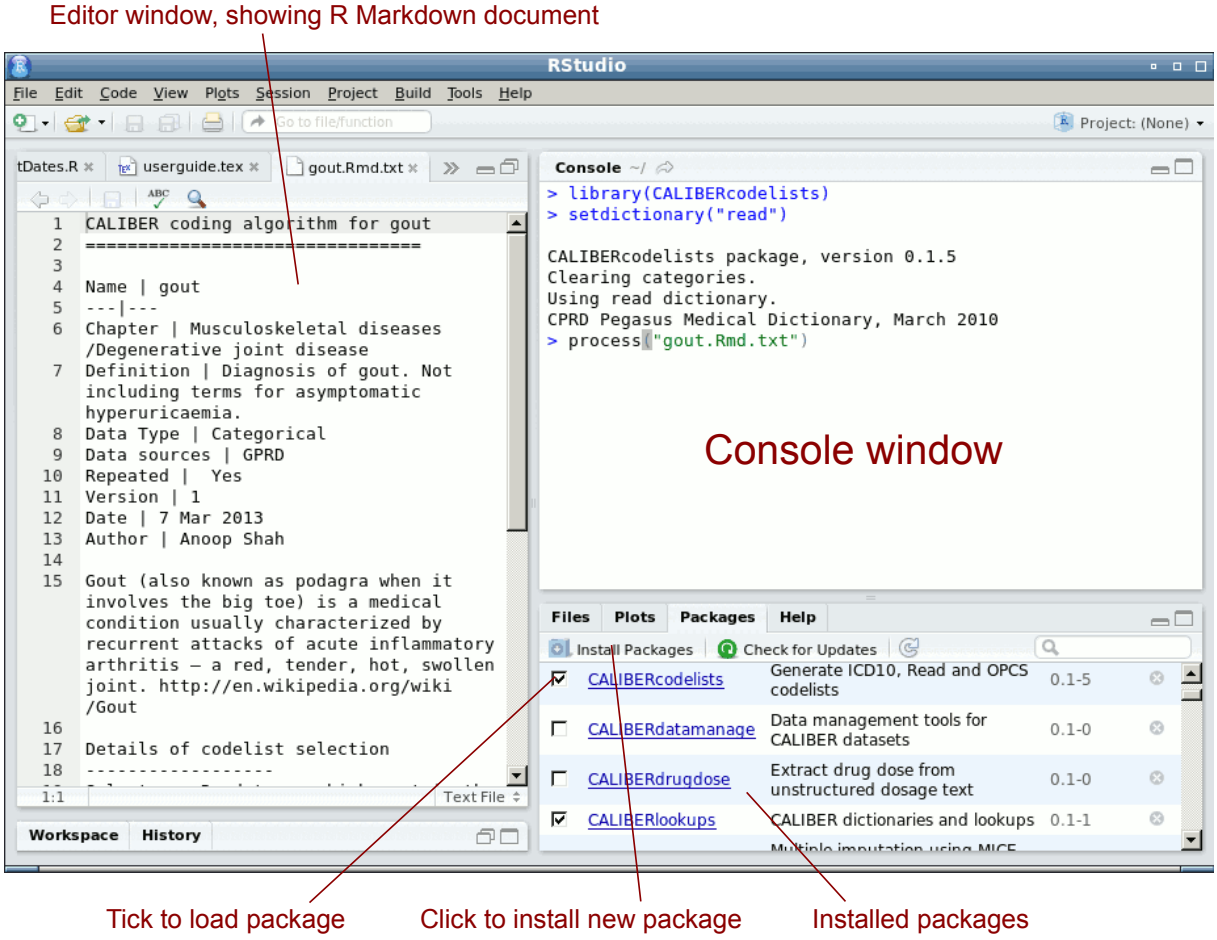
5 Installation

5.1 Installing R and RStudio for Windows

R is a console-based program like Stata, and there are a number of graphical user interfaces which can be used with it. This guide will be based on RStudio because it is free, user-friendly and available for Windows, Mac and Linux.

Download `R-2.15.3-win.exe` from `http://cran.r-project.org/`, and run it (ideally in Administrator mode) to install R on your computer. Then download RStudio from `http://www.rstudio.com/ide/download/` and install it. When you open RStudio for the first time it will either detect your computer's R installation or will ask you to find the `R` folder in `Program files` where R is installed.

Figure 1: Screenshot of RStudio



The RStudio workspace shows up to four panes (see Figure 1), which can be customised using the menu **Tools** → **Options** → **Pane Layout**. Some panes have alternative tabs to show different things, such as:

Source For editing R scripts (do-files), R Markdown documents (combining ordinary text and R code) or other text files. You can run a piece of R code by highlighting it and pressing **CTRL + Enter**

Console This is like the Stata Command and Results windows. You can type commands and see immediate output

Help Documentation for core R and packages you have installed

Packages List of installed packages

5.2 Installing and loading the CALIBERcodelists package

Download the **CALIBERcodelists_0.2-0.zip** file, which contains the installation for Windows. Type the following into the console:

```
install.packages(file.choose())
```

A file choose window will open, in which you should select the `CALIBERcodelists_0.2-0.zip` file. The computer should automatically install the packages `data.table`, `knitr` and `markdown`.

Each time you need to use the package, load it by typing in the console:

```
library(CALIBERcodelists)
```

Alternatively, you can install the package by clicking 'Install packages' in the **Packages** tab, then tick the box next to the package to load it (see Figure 1).

5.3 Obtaining the source dictionaries

The `CALIBERcodelists` package requires dictionaries of ICD-10, OPCS and Read codes. If these are not provided, the package will use a small test dictionary and display a message such as this:

```
SAMPLE READ DICTIONARY, for testing only.
Remove using the commands rm(list=c("CALIBER_DICT", "CALIBER_DICTMAPS"))
and re-run to load actual dictionary, if available.
```

5.3.1 CALIBER users

Install the `CALIBERlookups` package (in a similar way to installing the `CALIBERcodelists` package) to provide the lookup tables. This package does not need to be loaded using `library()` but just has to be present and installed on the system.

5.3.2 Non-CALIBER users

First, obtain the dictionary files listed below and place them in the R current working directory (the working directory can be changed using the command `setwd`, e.g. `setwd("C:/Documents/work")`, or using the RStudio menu **Session** → **Set Working Directory**).

Dictionaries from NHS terminology service

ICD-10 default filename 'ICD10_Edition4_CodesAndTitlesAndMetadata_GB_20120401.txt' (located within the Content folder of the NHS ICD-10 zip file nhs_icd10_1.0.0_20120401000001.zip). Tab separated. Column names: CODE, ALT_CODE, USAGE, USAGE_UK, DESCRIPTION, MODIFIER_4, MODIFIER_5, QUALIFIERS, GENDER_MASK, MIN_AGE, MAX_AGE, TREE_DESCRIPTION

OPCS default filename 'OPCS46 CodesAndTitles Jan 2011 V1.0.txt', supplied together with documentation in a zip file (nhs_opcs4_6.0.0_20110110000001.zip). Tab separated, no column headers, code in left column and term in right column.

Mapping ICD-10 ↔ Read default filename 'Icd10.v3', in the V3 folder of the Read release (supplied as a zip file e.g. nhs_readctv3_14.0.0_20121001000001.zip). This is a pipe-separated text file with no header row, and column order: 5-digit Read code, ICD-10 code, map status, ref flag, additional flag, element number, block number

Mapping OPCS ↔ Read default filename 'Opcs.v3', in the V3 folder of the Read release (supplied as a zip file e.g. nhs_readctv3_14.0.0_20121001000001.zip). This is a pipe-separated text file with no header row, and column order: 5-digit Read code, ICD-10 code, map status, ref flag, additional flag, element number, block number

Dictionaries from the Clinical Practice Research Datalink These dictionaries are supplied in GOLD format data (General Practice Research Database, GPRD-GOLD).

Pegasus Medical Dictionary default name 'pegasus_medical.txt'. This is a tab-separated file with metadata in the first two rows.

Pegasus Product Dictionary for completeness, the product dictionary is also processed by this script although it is not currently used by the `CALIBERcodelists` package. The default filename is 'pegasus_product.txt', and is also a tab-separated file with metadata in the first two rows.

Next, type `demo(loadnewdicts)` to run the load script.

The script will load the source dictionaries, create a subdirectory called 'data' and create lookup tables (as `.rda` R data files) called 'CALIBER_DICT.rda', 'CALIBER_PRODDICT.rda' and 'CALIBER_DICTMAPS.rda' in the `data` folder.

It should be possible to load the `CALIBER_DICT` dictionary using `data(CALIBER_DICT)`, and this is how the `CALIBERcodelists` package will load the dictionary. The `.rda` files need to be in the `data` subfolder of the current working directory whenever they are loaded using the `setdictionary` command.

If a lookup table is present in an installed package as well as the data subdirectory, the version in the package will be used in preference.

6 Codelists

6.1 What is a codelist?

A **codelist** is a set of Read, OPCS or ICD-10 codes with associated categories. Each code/term is only allowed to be in one category (i.e. the categories are mutually exclusive), and a single codelist can contain terms from only one of the source dictionaries. The categories should describe logical groupings of medical conditions that are useful for research. Each codelist is associated with information about itself called 'metadata', such as the version number, authors and category descriptions.

Codelists are used in **coding algorithms** to define CALIBER variables representing information about a patient, such as whether they have a particular disease.

The `CALIBERcodelists` package treats codelists as special objects, which are displayed in a particular way and have special functions to create and manipulate them. When you view a codelist object in R, it will look something like this:

```
Codelist based on read dictionary with 5 terms.

Name: hcm_gprd
Version: 1
Source: GPRD
Author: An author
Date: 11 Feb 2013
Timestamp: 15.02 11-Feb-13
Categories:
2.  FH: HCM
3.  HCM

TERMS (sorted by category and code):
```

```
   category    code                                          term medcode events
1:        2 12CJ.00                          FH: Cardiomyopathy   13274   2105
2:        2 12CR.00 FH: Hypertrophic obstructive cardiomyopathy   42999    540
3:        3 G551.00     Hypertrophic obstructive cardiomyopathy    8010   1909
4:        3 G554300 Hypertrophic non-obstructive cardiomyopathy    3499   1399
5:        3 Gyu5M00        [X]Other hypertrophic cardiomyopathy   70648     18
```

6.2 File format

The file format of the definitive codelist is a `.csv` (comma-separated values) file with the following columns:

metadata quoted text, containing Name: value pairs and the category table. The text is padded with spaces to ensure that the categories line up. The width of the column is determined by the length of the variable names and category levels.

category the category, right justified by adjusting the amount of right padding of the metadata column

icd_code, opcs_code or readcode quoted text

readterm, icd_term or opcs_term quoted text

medcode only for Read code lists, integer

events only for Read code lists, integer (optional)

These files can be opened in a spreadsheet or text editor, or imported into a statistical program. It is also possible to export to Stata format by specifying a filename ending in `.dta`. In this format, the 'category' column has value labels and the remainder of the metadata is in the 'datalabel' field, in the order: `name | version | date | author(s)`. When opened in Stata, the codelist looks like this:

```
. use hcm_gprd.codelist.1.dta
(hcm_gprd | 1 | 11 Feb 2013 | An author                                          )

. describe

Contains data from hcm_gprd.codelist.1.dta
  obs:             5                          hcm_gprd | 1 | 11 Feb 2013 |
                                                An Author
 vars:             5                          11 Mar 2013 16:15
 size:           330 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
category        long   %9.0g       category   category
code            str7   %7s                    code
term            str43  %43s                   term
medcode         long   %9.0g                  medcode
events          long   %9.0g                  events
-------------------------------------------------------------------------------
Sorted by:

. list category term medcode
```

```
    +-------------------------------------------------------------+
    | category                                          term  medcode |
    |-------------------------------------------------------------|
 1. |  FH: HCM                              FH: Cardiomyopathy   13274 |
 2. |  FH: HCM  FH: Hypertrophic obstructive cardiomyopathy   42999 |
 3. |      HCM      Hypertrophic obstructive cardiomyopathy    8010 |
 4. |      HCM  Hypertrophic non-obstructive cardiomyopathy    3499 |
 5. |      HCM          [X]Other hypertrophic cardiomyopathy   70648 |
    +-------------------------------------------------------------+
```

7 Overview of the codelist creation process

We divide the codelist creation process into the following steps (Figure 2):

1. Decide which source dictionaries to use (see subsection 9.1).
2. Create a **selection** of terms from the Read, OPCS or ICD-10 source dictionaries. For example, one might select all terms containing the word 'angina', or all ICD-10 terms beginning with 'I20'. Combine selections using Boolean operators such as AND, OR or NOT to identify exactly which terms are of interest (see subsection 9.2).
3. Allocate a **category** to terms in a particular selection. (see subsection 9.3)
4. Set the metadata for the codelists under construction (version number, category descriptions, author name, date). Extract the Read, ICD-10 and OPCS codelists seperately as **codelist** objects, and export them as `.csv` files (see subsection 9.4).
5. Ask clinicians and epidemiologists to review the codelists and suggest any changes. The selection algorithm can be changed and the results compared with the previous version if necessary. There are tools available to assist this process (see section 12).
6. Produce the final codelists along with a document describing how they were generated (see subsection 9.5).

The `CALIBERcodelists` package contains the source dictionaries and a 'codelist construction area', where codes to form a new codelist and metadata can be specified, and then combined into an exported file or a codelist object in the R workspace.

7.1 What is a selection?

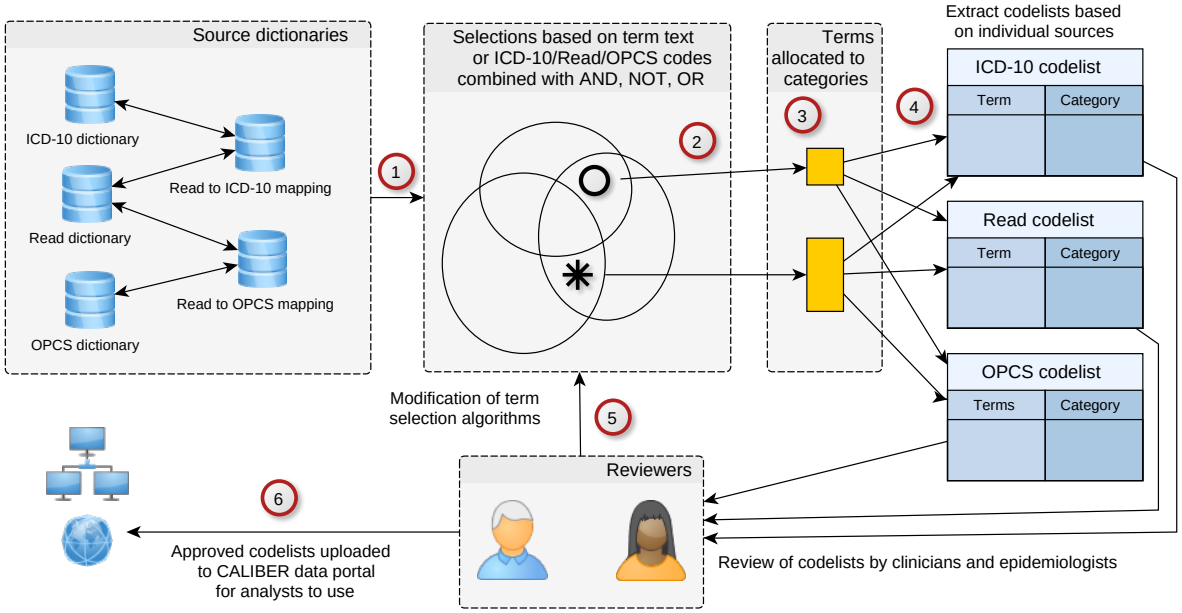
A **selection** is defined as a set of Read, OPCS and/or ICD-10 terms created by a selection procedure (e.g. searching on term text or code). A term can either be included in the selection or not; there are no categories. Selections are used in the process of building a codelist. A selection should be created to encompass all the codes for a particular category, combining selections with AND, OR and NOT as necessary.

For example, searching for terms matching the text 'hypertrophic' and 'cardiomyopathy' will yield this selection:

```
"hypertrophic" %AND% "cardiomyopathy"
```

```
Read terms:
   medcode    code                                          term events
1:   42999 12CR.00 FH: Hypertrophic obstructive cardiomyopathy    540
```

Figure 2: Overview of the codelist creation process




```
2:    8010 G551.00     Hypertrophic obstructive cardiomyopathy   1909
3:    3499 G554300 Hypertrophic non-obstructive cardiomyopathy   1399
4:   70648 Gyu5M00        [X]Other hypertrophic cardiomyopathy     18

ICD-10 terms:
   code                                    term
1: I421 Obstructive hypertrophic cardiomyopathy
2: I422       Other hypertrophic cardiomyopathy

OPCS terms:
Empty data.table (0 rows) of 2 cols: code,term
```

8 Creating a codelist using the interactive menu

The interactive scripts are invoked by the `demo` command, e.g. `demo(makecodelist)` or `demo(explore)`. Press ESCAPE to exit the demo at any time.

8.1 explore demo

This script provides facilities for managing existing codelists (e.g. converting them to a standardised format, comparing them with one another, merging two codelists) and creating a new codelist. The program presents a set of menus and options, allowing the user to type an option at each stage. All menu commands are converted into R statements and executed, and the relevant R code is displayed. A log is maintained of all the commands called, and on exiting, the output and commands are shown in the log file with everything except the commands commented out using #. This means that the log file is also an R script for re-running everything that was done in the interactive mode.

Pressing ESCAPE will abort the program without showing the log file; the log file (`interactive.log`) can be found in the current R temporary directory (type `tempdir()` to find out where it is).

The main menu is divided into two tasks: creating a new codelist and managing existing codelists.

Create new codelist

- Create or edit a selection → invokes the selection menu, allowing selection of codes and terms using regular expressions (see subsection 8.3)
- Assign a category to terms in a selection
- Edit metadata and category descriptions
- Export codelists

Manage codelists

- Load codelists from file
- Export codelists to file
- Convert codelist from one dictionary to another (subsection 12.4)
- Edit metadata and categories

Process R Markdown file

Convert codelist to standard format – Loads a codelist, asks the user for metadata and category names, and saves the codelist in a standardised format.

8.2 makecodelist demo

This script takes you through the process of creating an R Markdown file documenting the codelist creation process, via menu driven system. Markdown is a simple way of formatting text so that it can be converted to other formats such as HTML web pages. R Markdown allows you to include the R code within the document, so that it is all in one place.

8.3 Regular expressions

The POSIX regular expression syntax used in R is the same as that used by regexm (but not `strmatch`) in Stata, and is described in detail in a numbr of online manuals (e.g. `http://stat.ethz.ch/R-manual/R-patched/library/base/html/regex.html`).

By default, a regular expression can match any part of the target word. The following symbols have special meaning in regular expressions:

`.` – match any single chracter. For example, `f..t` will match 'foot' or 'feet'

`[]` – match any single character in the square brackets. For example, `co[ap]d` will match 'copd' or 'coad'. The following special characters can be used inside brackets:

`^` as the first character inside a pair of square brackets means that the character must not match anything in the brackets

`-` can be used to specify a range of letters or numbers

`|` – 'OR' operator. The pattern on either side of the pipe must match. Multiple pipes can be used to denote multiple options, e.g. `renal|kidney|nephro`

`\` – forces the next character to be treated 'as is' rather than as a special symbol, e.g. `\.` can be used to match `.`

Anchors These operators force the pattern to match the beginning or end of the target string:

^ – match the beginning of the string. For example, `^G` will match 'G30' but not '1G'. However, ^ as the first character inside a pair of square brackets means that the brackets must not match any of the characters in teh brackets

$ – match the end of the string. For example, `infarction$` will match 'splenic infarction' but not 'infarction of spleen'

Pipes have precedence over anchors, so `^heart|heart$` will match texts either starting or ending with the word 'heart'.

Number These operators allow the previous letter, number, symbol or set of square brackets to be repeated a particular number of times:

(nothing) – match exactly once (default)

+ – match 1 or more times. For example, `[ ]` will match one or more spaces

* – match 0 or more times. For example, `.*` will match any number of any type of character, so `myo.*infarct` will match 'myo infarct' and 'myocardial infarction'

Brackets can be used to set an order of precedence, for example:

`(heart attack)|(myocardial|anterio|inferior).*infarct` will match 'heart attack', anterior infarct', 'myocardial infarct' or 'inferior infarct', but not 'heart infarct'.

Examples of regular expressions

Stata strmatch	Regular expression	Phrases matched	Phrases not matched
`A*`	`^A`	AB, A1	BA, Z
`A?B*`	`^A.B`	A.B2, A1B	AB, BA, A..B
`A.*`	`^A\\.`	A.1, A.B	A1, BA, B1
`*stab*angina*`	`stab.*angina`	stable angina	angina stable
	`stab.*angina|heart`	heart, unstable angina	angina stable
	`hosp|^admit`	in hospital, hospitalise, admit	not admitted
	`f[oe]+t`	foot, feeet, foooot	ft, fleet
	`h[^0-9]r`	har, h.r, her	h0r, h7r, hear
	`h[^0-9]*r`	har, h.r, her, hear	h0r, h7r

8.4 The menu system

8.4.1 Main menu

The main menu is invoked using `demo(explore)`. It proves the following options:

1 Use existing codelist (in memory or from file) – see subsubsection 8.4.2

2 Create new codelist – see subsubsection 8.4.3

3 Exit

It is possible to exit at any stage by pressing ESCAPE but the history will not be displayed if this happens, so it is recommended to exit using the menu.

8.4.2 Use existing codelist

This menu shows a list of codelists currently loaded in the R workspace, of which one is currently 'selected'. The following options are available:

1 View selected codelist – shows the codelist on screen

2 Browse and edit in spreadsheet – allows the categories to be modified using a spreadsheet

3 Merge with another codelist and overwrite categories – if codes are in both codelists

4 Edit metadata – edit the name, author, date and version

5 Edit categories – edit the category labels

6 Compare to an older codelist

7 Save to disk – saves it as a CSV text file or Stata file

8 Convert to another dictionary using NHS mapping

9 Clear from memory – removes the codelist from the R workspace, but does not delete the original file if it was loaded from file

10 Select or load another codelist

11 Exit to main menu – see subsubsection 8.4.1

8.4.3 Create new codelist

1 Reset categories and choose dictionaries

2 Create a new selection – creates an initial selection based on terms or codes, then invokes the create selection menu, see subsubsection 8.4.4

3 Use an existing selection

4 Assign a selection to a category

5 Edit categories

6 Edit codelists under construction in spreadsheet – this is also the only way to view the codelists under construction

7 Compare to an older codelist

8 Edit metadata

9 Save/export new codelists and exit to main menu – allows codelists to be stored in memory as R objects and also saved to disk. Returns to the main menu (subsubsection 8.4.1).

8.4.4 Create selection menu

1 View in console – shows the selection on screen. If there are too many terms, it is better to use option 2 and browse it as a spreadsheet

2 Browse in spreadsheet – allows you to view the selected terms and delete any that you wish to remove from the selection. However it is generally better to remove terms based on the text or code rather than a list of individual terms.

3 Add terms by term text

4 Add terms by icd10/opcs/read code

5 Add terms from another selection – results in the union of the two selections

6 Limit selection by term text

7 Limit selection by icd10/opcs/read code

8 Limit selection using another selection – results in the intersection of the two selections

9 Limit to one source dictionary

10 Remove terms by term text

11 Remove terms by icd10/opcs/read code

12 Remove terms using another selection

13 Assign category to terms in selection – asks for a name for the new selection, stores it in memory and assigns the codes to the new category. Then returns to the create codelist menu (subsubsection 8.4.3)

14 Undo last operation – to undo the most recent operation on this selection

15 Store in memory and exit to CREATE CODELIST MENU – asks for a name for the new selection, stores it in memory and returns to the create codelist menu (subsubsection 8.4.3)

9 Creating a codelist using R Markdown

The **makecodelist** demo (subsection 8.2) is an interactive way of creating an R Markdown document combining code and descriptive text. This section describes how to create such a document from scratch, and then process it to create the codelist and documentation.

Before starting to code, it is important to have a general idea of possible search terms to search for. This may require literature review, discussion with subject experts or looking in a hierarchical browser such as the WHO online ICD-10 browser.

Ensure that the `CALIBERcodelists` package is loaded:

```
library(CALIBERcodelists)
```

Create a directory for the documents in this new codelist (in this example we will create a e.g. `N:/Documents/hcm/`). Create a new R Markdown document (with .Rmd extension) and save it in the codelist folder; you can do this from the menu in RStudio. Also change your current working directory to the codelist directory, either using the menu (Session → Set Working Directory) or using the `setwd()` function, e.g.:

```
setwd("N:/Documents/hcm/")
```

(Filepaths must always be quoted, unlike Stata.)

We can now create the first part of the Markdown document, containing the metadata and description. In the HTML document the metadata will appear in a table, but this information will also be include in the exported codelist.

```
Codelist for hypertrophic cardiomyopathy
========================================================

Name    | hcm_demo
--------|-----------
```

```
Source | GPRD, HES
Date   | 11 Feb 2013
Author | Author1, Author2, Author3
Version| 1

Description
-----------
A codelist containing Read terms for hypertrophic cardiomyopathy
and family history of cardiomyopathy.
```

The basic rules of R Markdown are as follows:

- Markdown documents are text documents and can be read and edited in any text editor (e.g. Notepad or the RStudio editor).
- Line breaks are ignored; leave a blank line to start a new paragraph
- Use `======` under main headings and `-----` under subheadings
- Create tables using vertical pipes (|) between the columns, and a line of hyphens (-) after the first row.

For more information follow this link: `http://www.rstudio.com/ide/docs/authoring/using_markdown`

Now we can create the code to select the terms. We can try out small sections of code by typing them in the Rmd document, highlighting them and pressing **CTRL + Enter**, or alternatively typing them straight into the console. To enter the R code in the Markdown document, it must be demarcated as follows:

````
(ordinary text)

```{r}
# Put your R code here
```

(ordinary text)
````

(The ‘ symbol is the top left key on a UK keyboard.)

9.1 Specifying which dictionary to use

First we need to clear any previous data and select which dictionaries to use in this session. Usually one would use Read and ICD-10 for diagnoses, and Read and OPCS for procedures.

```
setdictionary("read", "icd10")
```

Note that this applies to the entire session. (To limit a single selection to a particular dictionary, use the `dictis` function.)

9.2 Creating selections

We can create a selection of terms using one or more of the following functions:

termhas select by term text using regular expressions

codematch select by codes from a particular dictionary, and include mapped terms in other dictionaries

dictis select all the terms in one of the three dictionaries

%OR% combine selections using OR (i.e. the union), and convert their arguments to selections using **termhas** if they are not already selections

%AND% combine selections using AND (i.e. the intersection), and convert their arguments to selections using **termhas** if they are not already selections

NOT negates a selection, converting its argument to a selection using **termhas** if it is not already a selection

termhas: selecting by term text

This function has the following 'Usage' entry in the help documentation:

```
termhas(regexpr, exact = FALSE, ignorecase = TRUE)
```

This means that regexpr (the search text) is a mandatory argument and the others are optional. The default exact = FALSE means that regexpr is interpreted as a regular expression (**??**) rather than having to match the term exactly, and ignorecase = TRUE means that it is not case sensitive.

We could look for family history of cardiomyopathy terms in Read like this:

```
termhas("cardiomyopathy") %AND% termhas("family|fh:")
```

We can abbreviate this by removing the 'termhas' function calls, as **%AND%**, **%OR%** and **NOT()** will automatically invoke `termhas` if the argument is a text string. (If we were using a single search term without %AND%, it would be necessary to use `termhas`).

We can save the selection for later use by naming it using the assignment operator <-:

```
myselection <- "cardiomyopathy" %AND% "family|fh:"
```

```
myselection
```

```
Read terms:
   medcode    code                                          term events
1:   13274 12CJ.00                          FH: Cardiomyopathy   2105
2:   42999 12CR.00 FH: Hypertrophic obstructive cardiomyopathy    540

ICD-10 terms:
Empty data.table (0 rows) of 2 cols: code,term
```

If you want to browse the entries with a particular search term or contained in a particular selection object, use the **browseSelection** function, e.g. `browseSelection("meningitis")` or `browseSelection(myselectio`
This extracts the relevant section of the dictionary, writes it to a temporary file and opens it in a spreadsheet program (default OpenOffice/LibreOffice on Linux and Excel on Windows). This can be useful for viewing which terms would be selected using a particular search strategy.

Further examples:

`termhas("meningitis")` – selects terms containing 'meningitis'

`NOT(termhas("meningitis"))` – terms which do not match 'meningitis'

`NOT("meningitis")` – shorthand for `NOT(termhas("meningitis"))`. NOT() coerces its argument to a selection using **termhas** if it is not already a selection.

codematch: selecting by Read, ICD-10 or OPCS code

This function has the following arguments:

```
codematch(regexpr, dictionary, mapStatus = NULL, exact = FALSE)
```

The first argument is a regular expression matching the code. The dictionary argument must be one of "read", "icd10" or "opcs", lower case and quoted exactly as in this text. Unlike **termhas** (section 9.2), **codematch** *is* case sensitive.

We know that the ICD-10 codes for hypertrophic cardiomyopathy are I421 and I422, so we can search for them like this:

```
codematch("I42[12]", dictionary="icd10")
```

```
Read terms:
   medcode    code                                          term events
1:    8010 G551.00     Hypertrophic obstructive cardiomyopathy   1909
2:    3499 G554300 Hypertrophic non-obstructive cardiomyopathy   1399
3:   70648 Gyu5M00        [X]Other hypertrophic cardiomyopathy     18

ICD-10 terms:
   code                                    term
1: I421 Obstructive hypertrophic cardiomyopathy
2: I422       Other hypertrophic cardiomyopathy
```

It is frequently useful to anchor the search pattern to the beginning of the code, particularly for Read codes, e.g. `codematch("^A", dictionary="read")` will find all Read codes beginning with 'A'.

9.3 Assigning terms to categories

Once we have created the desired selections, we can assign categories to them using the `assigncat` function:

```
assigncat(2, "FH: HCM|Family history of cardiomyopathy", myselection)
```

We could alternatively type the code to create the selection in the assigncat function instead of the named selection object 'myselection', and this might be more convenient if it is not too long, e.g.:

```
assigncat(2, "FH: HCM|Fam hx of cardiomyopathy", codematch("I42[12]", "icd10"))
```

```
assigncat(1, "Meningitis", "meningitis" %AND% NOT("vaccination"))
```

The first argument is the category number (do not use negative numbers, and use 0 only for codes to be excluded), the second is the category description (with an optional short name separated by |), the third is the name of a **selection** object or an expression to create a selection object.

If you assign a term to a category, it will overwrite the category which was previously assigned to that term. The optional cats_to_convert argument can be specified if you want to convert only specific categories. The argument should be supplied as a vector of numbers, which is created using the c (concatenate) function, or by specifying numerical ranges such as 1:4. Example:

```
assigncat(0, "Exclusions", termhas("family|fh:"), cats_to_convert=c(1:3, 6))
```

9.4 Exporting codelists

To finish off, if we want to save the codelist to disk we need to include an `exportall` command in the R markdown document. The optional argument to this function is the directory in which to

place the new codelists, which defaults to the same directory as the R Markdown document if not supplied. Codelists are automatically named using the format:

Name_source.codelist.**Version**.csv

where **Name** is the **Name** attribute, **source** is the data source ('gprd', 'hes' or 'opcs') and **Version** is the version number (the **Version** attribute).

For example, version 1 of the GPRD (Read) AF codelist would be exported to a file named

`af_gprd.codelist.1.csv`

All codelists in the CALIBER data portal (`http://www.caliberresearch.org/portal/`) will follow this naming scheme.

Codelists can also be exported individually using the `export` function, which allows the name to be specified. If the file extension is `.dta`, it is exported in Stata format (see subsection 6.2).

9.5 Processing the R Markdown file

The 'code' section of our example Markdown document looks like this:

```
```{r}
setdictionary("read", "icd10")
hcm <- ("hypertrophic" %AND% "cardiomyopathy") %OR%
  codematch("I42[12]", dictionary="icd10")
assigncat(3, "HCM|Hypertrophic cardiomyopathy", hcm)
assigncat(2, "FH:HCM|Family history of cardiomyopathy",
"cardiomyopathy" %AND% "family|fh:")
exportall()
```
```

To run the whole codelist selection process type this command in the console:

```
process("hcmdemo.Rmd", showR=FALSE, showhtml=TRUE, show_not_selected=FALSE)
```

or to use the default options (don't show R code, but show HTML codelist and unselected terms):

```
process("hcmdemo.Rmd")
```

This will produce the HTML document (Figure 3) and export the codelists to the definitive `.csv` format. If a codelist of this name already exists in the folder (e.g. a previous version that you were working on), the differences between the current codelist and the previous version will be displayed.

10 Creating codelists using Stata

Codelists can be created using Stata as follows:

1. Load the dictionary. If using the `CALIBERlookups` package, the dictionary `CALIBER_DICT` can be exported from R as follows:
   ```
   library(CALIBERlookups)
   data(CALIBER_DICT)
   write.csv(CALIBER_DICT, " (filename) ", row.names=FALSE)
   ```

2. Reset the category column to missing

Figure 3: HTML documentation of coding algorithm

Codelist for hypertrophic cardiomyopathy

Name	**hcm_demo**
Source	GPRD, HES
Date	11 Feb 2013
Author	Author1, Author2, Author3
Version	1

Description

A codelist containing Read terms for hypertrophic cardiomyopathy and family history of cardiomyopathy.

Codelists for variable

Generated on 2013-03-11 17:49:49 using the CALIBERcodelists package, version 0.1.5 and source dictionaries:

ICD-10 codes, terms and text used by permission of WHO, from: International Statistical Classification of Diseases and Related Health Problems, Tenth Revision (ICD-10). Vols 1-3. Geneva, World Health Organization, 1992-2000. 4th Edition.

CPRD Pegasus Medical Dictionary, March 2010

Categories:

Category	Short name	Description	Number of terms
2	Family history of cardiomyopathy	FH:HCM	2
3	HCM	Hypertrophic cardiomyopathy	5

Note: The number of terms for ICD-10 includes all 5-character sub-codes and may be greater than the number of terms shown below

Terms included in variable definition

Category 2: FH:HCM

Read terms

code	term	category	events
12CJ.00	FH: Cardiomyopathy	2	2105
12CR.00	FH: Hypertrophic obstructive cardiomyopathy	2	540

ICD10 terms

(no terms)

Category 3: Hypertrophic cardiomyopathy

Read terms

code	term	category	events
G551.00	Hypertrophic obstructive cardiomyopathy	3	1909
G554300	Hypertrophic non-obstructive cardiomyopathy	3	1399
Gyu5M00	[X]Other hypertrophic cardiomyopathy	3	18

ICD10 terms

code	term	category
I421	Obstructive hypertrophic cardiomyopathy	3
I422	Other hypertrophic cardiomyopathy	3

Codes and terms initially selected then excluded in variable definition

(no terms)

3. Use the Stata `regexm` command to use regular expression searching on term or code, and assign categories, e.g.
   `replace category = 1 if regexm(term, "angina|Angina")`
4. Keep only the rows with positive categories, e.g.
   `drop if category == . | category < 1`
5. Generate value labels for the categories.
6. Save the dataset as a Stata DTA file.

Codelists can be created using Stata and then converted into a standard format using the 'Standardise codelist' option in `demo(explore)`. This menu asks the user to enter metadata such as author name and version, and saves the codelist in the standardised CALIBER format.

11 Drug (product) codelists

Functions for manipulating drug codelists have not been coded using the interactive system, as these codelists are less commonly used. Therefore standard R data.table code needs to be used to create them.

The source dictionary is `CALIBER_PRODDICT`, available in the `CALIBERlookups` package.

11.1 Working with data.table objects

The master dictionary is a data.table, and data.table objects have special functions to generate subsets and update them, rather like using SQL to update a database. These commands are generally written in square brackets after the name of the data.table.

```
name_of_data_table[i]
```

This selects the rows of the data.table. For example, you can use `grepl` to select and view selected rows of the data.table by searching on text. The first argument to `grepl` is the regular expression to be matched, the second is a character vector in which to search.

```
CALIBER_PRODDICT[grepl("ivabradine", drugsubstance)]
```

```
CALIBER_PRODDICT[grepl("SYMBICORT", prodname)]
```

Data.table objects can also be used with a `j` argument, which is an expression to be evaluated and returned for each row. If `j` is not supplied, all columns are returned by default. The `j` argument can either be a selection argument which does not modify the data.table, or an update expression using the `:=` operator.

These commands return specified columns for the rows which which have 'SYMBICORT' in the product name.

```
CALIBER_PRODDICT[grepl("SYMBICORT", prodname), prodname]
```

```
[1] "SYMBICORT TURBOHALER 200micrograms + 6micrograms/actuation [ASTRAZENEC]"
[2] "SYMBICORT TURBOHALER 400micrograms + 12micrograms/actuation [ASTRAZENEC]"
[3] "SYMBICORT TURBOHALER 100micrograms + 6micrograms/actuation [ASTRAZENEC]"
```

(result is a simple vector)

```
CALIBER_PRODDICT[grepl("SYMBICORT", prodname), list(prodname)]
```

```
                                                                    prodname
1:  SYMBICORT TURBOHALER 200micrograms + 6micrograms/actuation [ASTRAZENEC]
2: SYMBICORT TURBOHALER 400micrograms + 12micrograms/actuation [ASTRAZENEC]
3:  SYMBICORT TURBOHALER 100micrograms + 6micrograms/actuation [ASTRAZENEC]
```

(result is a data.table with one column)

```
CALIBER_PRODDICT[grepl("SYMBICORT", prodname), list(prodcode, multilex)]
```

```
   prodcode multilex
1:     6325 09605002
2:     6780 08506001
3:     7013 09605001
```

The next command deletes the 'category' column (if it exists) and then assigns rows containing 'SYMBICORT' to category 1 (the other values in the category column remain as they are; in this case missing (NA)).

```
CALIBER_PRODDICT[, category:=NULL]
```

```
CALIBER_PRODDICT[grepl("SYMBICORT", prodname), category:=1]
```

Missing values cannot be selected using ==. Instead the function is.na() needs to be used to return a TRUE / FALSE vector for whether each value of a vector is missing. The ! operator is used for negation, so to select all rows with non-missing categories, use the command:

```
CALIBER_PRODDICT[!is.na(category)]
```

11.2 Creating a codelist from the product dictionary

First, load the package and the product dictionary:

```
library(CALIBERlookups)
```

```
data(CALIBER_PRODDICT)
```

The following fields may be useful:

prodname product name

drugsubstance active ingredient in the product

strength e.g. 2.5mg

formulation e.g. tablets

route e.g. Oral

bnfcode BNF chapter number, with multiple entries separated by / if the drug belongs to more than one chapter (e.g. 02090000/04070100/10010100)

bnfheader BNF header text

Create a category column as described in the previous subsection, then select the rows of CALIBER_PRODDICT with non-missing, non-zero categories.

```
mySymbicortCodelist <- as.codelist(CALIBER_PRODDICT[!is.na(category) & 0 < category])
```

The Source attribute for product codelists is 'GPRDPROD', and codelist files are suffixed:

_gprdprod.codelist.<version>.csv

Product codelist can now be viewed, exported and compared using the same functions as Read, OPCS or ICD-10 codelists.

12 Utility functions

12.1 Importing codelists into a standard format

The **as.codelist** function converts a file or another object into a codelist object. The argument can be one of the following:

- A text file with columns medcode, icd_code or opcs_code and a category column (which can have a name other than 'category'). This covers the format of most existing CALIBER codelists.
- A Stata `.dta` file with similar format to a text file.
- A data.frame or data.table in a similar format to a text file.
- The name of a source dictionary ('read', 'icd10' or 'opcs'), in which case the codelist under construction is extracted.
- A selection (need supply the `dictionary` argument if the selection contains terms from more than one dictionary). The default category is set to 1 unless specified.

12.2 Editing codelists

Codelists in R are an instance of the S3 object 'codelist', implemented as a data.table with columns:

code Read, ICD-10 or OPCS code (named 'readcode', 'icd_code' or 'opcs_code' in the exported codelist)

term Read, ICD-10 or OPCS term (named 'readterm', 'icd_term' or 'opcs_term' in the exported codelist)

category category number for the term

medcode (Read codelists only) the medcode relating to the Read code in the CPRD-GOLD data format

Unlike most objects in R, data.table and codelist objects are updated by reference, and in order to copy an object you need to use the `copy` function explicitly.

`codelist2 <- codelist1` will create a new alias 'codelist2' which points to the same underlying object as codelist1.

`codelist2 <- copy(codelist1)` creates a new codelist which is cloned from codelist1, but any subsequent changes to codelist2 will have no effect on codelist1.

To create a subset of a codelist, use the `subset` function which takes as arguments the original codelist and an expression stating which rows to keep, e.g.:

`codelist2 <- subset(codelist1, category==2)` to create a new codelist containing only category 2 from the original codelist.

Categories for individual terms can be altered using the data.table update operator (`:=`), writing L after the category number to specify that it is an integer, e.g.:

```
codelist1[term=="Stable angina", category:=2L]
```

It is also possible to edit the categories in a spreadsheet, using `browseCodelist(codelist1)`. This function opens the codelist in a temporary spreadsheet in Excel or LibreOffice. The categories can be manually edited, and if the spreadsheet is saved under the same name it can be automatically reloaded into R to update the categories in the R object.

Metadata are stored as `attributes` of the object, and can be set using **setMetadata**, which takes a codelist as the first argument and attributes as other named arguments e.g.:

```
setMetadata(mycodelist, Name="angina_hes")
```

The following metadata apply to every codelist:

Name name of the codelist, suffixed by _gprd, _hes, _ons, _opcs or _gprdprod in the exported codelist. However, do not use the suffix if creating codelists for multiple dictionaries simultaneously (e.g. ICD-10 and Read) because they will be added automatically.

Author names of author(s) of the codelist

Version use version numbers less than 1 for codelists that have not been approved. Final versions should be numbered 1, 2, etc. The version is part of the codelist filen. tered date the codelist was developed

Timestamp date the codelist was exported, should not be modified by the user

Source GPRD, HES, ONS, OPCS or GPRDPROD; assigned automatically. ICD-10 codelists default to HES but this can be changed using the `setMetadata` function.

Categories data.table with columns: category (integer), shortname (character), description (character). Categories with numbers less than 1 are not exported. Category 0 is to be used only for excluded terms, and -1 is used internally by the `CALIBERcodelists` package. All coding categories must be positive integers.

For the codelist under construction using the master dictionary, the metadata are stored in the hidden `META` table. Metadata can be assigned by calling `setMetadata` without a codelist, e.g. `setMetadata(Name="angina")`.

The master category table is accessed via the functions `addCategory`, `retrieveCategories` and `saveCategories`.

12.3 Comparing and merging codelists

`compare(oldlist, newlist)` (where newlist is the master dictionary if null) lists the differences between terms with positive categories in the two codelists.

Whenever a codelist is generated by processing an R markdown document, any codelist exported is automatically checked against an existing codelist of the same name in the same location (if it exists), and the results are displayed in the console.

`merge(x, y)` creates a new codelist containing all terms in codelist `x` or `y`, with categories from codelist `y` over-writing those in codelist `x` if a term is present in both codelists. This may be useful to consolidate sets of codes selected by two different processes or individuals.

12.4 Converting codelists between Read and ICD-10, and between Read and OPCS

Conversion of codelists using the **convert** function makes use of the NHS mapping. This is not considered robust enough for research use without further manual review of the codes, but may be useful to create an initial approximate Read codelist from ICD-10 with little effort.

It may not be possible to convert a codelist with many categories if terms in different categories in the source codelist are mapped to the same term in the target dictionary. In such cases it would be necessary to split the codelist into individual categories, convert each one separately and then manually review the maps and resolve any conflicts.

12.5 Contracting and expanding ICD-10 codelists

`contractCodelist(codelist)` and `expandCodelist(codelist)` can be used on ICD-10 codelists to group or ungroup 4-character ICD headers. By default, the 3-character ICD headers are not used for selecting terms of interest because they are not permitted for use in HES coding. However, in the exported codelist, codes are in the 'contracted' format, grouped by 3-character ICD-10 header, which reduces the number of codes and makes them easier to understand and use in SQL. An example using a simple codelist:

```
mi <- as.codelist(codematch("I21", "icd10"), dictionary="icd10")
```

```
expandCodelist(mi)
```

```
   code                                                  term hierarchy category
1:  I21                             Acute myocardial infarction    parent        1
2: I210 Acute transmural myocardial infarction of anterior wall     child        1
3: I211 Acute transmural myocardial infarction of inferior wall     child        1
4: I212   Acute transmural myocardial infarction of other sites     child        1
5: I213 Acute transmural myocardial infarction of unspecifie...     child        1
6: I214              Acute subendocardial myocardial infarction     child        1
7: I219                Acute myocardial infarction, unspecified     child        1
```

```
contractCodelist(mi)
```

```
   code                        term category
1:  I21 Acute myocardial infarction        1
```