bib2gls: Standalone entries and repeated lists (a little book of poisons)

Nicola L. C. Talbot

Abstract

Most articles that describe how to use the glossaries package consider a single sorted list or possibly multiple lists where each list has a different set of entries (terms, symbols, abbreviations, etc.). However, some documents may instead have each term described in the main matter, with references to the term linking back to that point in the document rather than to a summary list. Alternatively (or additionally) a document may have multiple lists consisting of the same set of entries ordered in different ways.

The examples here were compiled with glossaries v4.49 [9], mfirstuc v2.07 [10], glossaries-extra v1.47 [8] and bib2gls v2.8 [6]. Some features and commands are not available in earlier versions. Also some earlier versions have bugs causing unexpected results.

1 The Book of Poisons

The Book of Poisons by Stevens and Bannon [3] is an excellent guide for crime fiction writers. It also provides a good example of a mixture of standalone entries (where each entry, in this case a toxic substance, is described in the main matter rather than in a summary list) and repeated lists in the back matter with just the name and no description (ordered by method of administration, by form, by symptoms, by toxicity and by time taken to react). It also has the more traditional lists (a bibliography and a simple glossary of medical terms) which aren't under discussion here. Finally, there's an index, which could be implemented using the standard `\index` and associated commands, but since it's possible to create the index as a by-product of using `bibgls` for the standalone entries, this will also be covered here.

The book consists of numbered chapters for each particular type of poison (household chemicals, plants, animals and so on). Each chapter is divided into unnumbered (sub) sections describing each poison, using a consistent structure listing:

Scientific Name (optional) for example 'Cantharis vericatoria' is the scientific name of cantharidin;

Other Name (optional) for example 'Spanish fly' is a colloquial term for cantharidin;

Other Similar (optional) a list of similar substances that don't have their own entries; for example, 'choline' is listed as similar to aspirin;

Toxicity a number from 1 (low) to 6 (highly toxic);[1]

Form/Deadly Parts plants have a deadly parts item (identifying which part of the plant is toxic), while other entries list the form (for example, liquid or gas) that the toxin takes;

Effects and Symptoms such as headache or nausea;

Reaction Time how long for symptoms to occur;

Antidotes and Treatments whether or not an antidote is available or if there is known treatment;

Notes (optional) some additional information, sometimes including a case history.

I don't want to bog down the examples with unrelated style code, so I'm just going to use the standard `description` environment. The entry data itself will also be significantly pared down to the following, which will correspond to glossary entry fields (custom ones will need to be defined):

`name` the toxin name (as it will appear in the section title);

`description` information about the toxin;

`toxicity` (custom field) a number;

`method` (custom field) the method of administration, a list of one or more elements from the set: breathed, injected, membrane absorption, skin absorption, smoked, swallowed.

`symptom` (custom field) the symptoms, which should be a list of one or more elements from any of the symptom classification subsets, including:

- **vital signs:** bradycardia, hyperthermia, hypertension, hypotension, hypothermia, tachycardia;
- **head, eyes, nose, throat:** bad/unusual taste, etc.

(There are too many to list individually here.)

I've omitted form (aerosol etc.) as the 'Ordered by Form' list can be achieved in the same way as the 'Ordered by Method' list. Similarly for the reaction time, which could be implemented with a numeric identifier like the toxicity.

2 Manual method

I'm going to start with an example document that doesn't use the glossaries package. Since we live in a digital age where some people prefer to read books on their devices, I've used the hyperref package [2]. The chapters are numbered in the main matter, but sections aren't numbered anywhere. The sections and subsections are too numerous to list in the table

[1] The toxicity level relates to the amount required for a lethal dose. It's not a measure of symptoms. A level 6 toxin can cause death from a pinprick amount with few symptoms, whereas a level 2 toxin requires a much higher dose to kill but can have debilitating long-term symptoms from a non-lethal dose.

doi.org/10.47397/tb/43-1/tb133talbot-bib2gls-reorder

of contents, but they would be useful in the PDF bookmarks. This can be achieved by setting the tocdepth counter to 0 (to only show chapters in the table of contents), setting the secnumdepth counter to 0 (to only show numbers for chapters) and using hyperref's bookmarksdepth option to set the depth for the PDF bookmarks.

The hyperref package automatically creates an anchor at the start of each page where the anchor name is obtained from the formatted page number. The `\frontmatter` command resets the page counter to 0 and changes the page number format to lowercase Roman numerals. Thus, the first page of the table of contents is 'i' and so the anchor for that page is `page.i`. The `\mainmatter` command similarly resets the page and changes the format to arabic (the default page number format) so the anchor for the first page of the first chapter will be `page.1`.

Unfortunately, the title page (and its reverse) also use the default number format so, even though the numbering may be hidden by the empty page style, the page anchor is still created (`page.1` for the title page and `page.2` for its reverse). Since the page numbers are hidden, the simplest solution is to select a different number format that isn't used for any of the other pages. In this case, I've used the `alph` format. This means the first two pages have the anchors `page.a` and `page.b`. They're not required anywhere in the document but this prevents a conflict and ensures that any references to pages 1 or 2 in the index or glossaries (once they are added) link to the correct page.

The 'Ordered by' lists mostly have very narrow columns so I've used the multicol package [1]. As we'll see, the 'Ordered by Toxicity' listing rounds 4.5 down to 4, which is what Stevens and Bannon do (although their toxicity list is in the opposite order).

Some of the scientific names are New Latin names, so I've provided a semantic command to typeset them:

```
\newcommand*{\latinname}[1]{\emph{#1}}
```

This example document contains only one toxin (ammonia), but it's already lengthy as it has a large number of symptoms.

```
\documentclass{book}
\usepackage{multicol}
\usepackage[bookmarksdepth=2]{hyperref}
\title{A Little Book of Poisons}
\author{Ann Author}
\setcounter{secnumdepth}{0}
\setcounter{tocdepth}{0}
\newcommand*{\latinname}[1]{\emph{#1}}
\begin{document}
\pagenumbering{alph}\pagestyle{empty}
\maketitle
\frontmatter\pagestyle{headings}
\tableofcontents
\mainmatter
\chapter{Household Poisons}
\section{Chemicals}
\subsection{Ammonia}
\begin{description}
\item[Other] Ammonium hydroxide.
\item[Toxicity] 4.5
\item[Method] Breathed.
\item[Symptoms] Tachycardia, blindness, lip/mouth
irritation, burns, flushing, coughing, pulmonary
edema, abdominal or stomach pain, restlessness,
collapse, and pain.
\item[Description] Some information about ammonia.
\end{description}
\backmatter
\chapter{Ordered by Administration}
\begin{multicols}{3}
\section{Breathed}
\begin{itemize}
\item[] Ammonia
\end{itemize}
\end{multicols}
\chapter{Ordered by Symptoms}
\begin{multicols}{3}
\section{Vital Signs}
\subsection{Tachycardia}
\begin{itemize}
\item[] Ammonia
\end{itemize}
% Lots of other sections omitted for brevity
\end{multicols}
\chapter{Ordered by Toxicity}
\begin{multicols}{3}
\section{Toxicity Rating 6}
\section{Toxicity Rating 5}
\section{Toxicity Rating 4}
\begin{itemize}
\item[] Ammonia
\end{itemize}
\section{Toxicity Rating 3}
\section{Toxicity Rating 2}
\end{multicols}
\end{document}
```

3 Standalone entries

For this next example, I'm going to consider a cut-down version of the main matter in order to illustrate standalone entries. The simplest approach can be achieved with the base glossaries package, although this has limitations. For now, each entry just has a name, toxicity and description. The toxicity could be stored in one of the custom keys, such as `user1`, but I've decided to define a new key called `toxicity`:

```
\glsaddstoragekey{toxicity}{}{\toxicity}
```

This both defines the `toxicity` key and provides a command called `\toxicity` to access the value.

Since the descriptions are likely to be quite lengthy and may contain paragraph breaks, they are best defined with `\longnewglossaryentry`:

```
\longnewglossaryentry{ammonia}
  {name=ammonia,toxicity=4.5}
  {Some information about ammonia.}
\longnewglossaryentry{nutmeg}
  {name=nutmeg,toxicity=3}
  {Some information about nutmeg.}
\longnewglossaryentry{lsd}
  {name=LSD,toxicity=2}
  {Some information about LSD that includes
  a reference to nutmeg.}
\longnewglossaryentry{botulinum}
  {name={botulinum},toxicity=6}
  {Some information about botulism.}
```

These are all defined in the file `toxins.tex`, which needs to be input in the preamble (with either `\input` or `\loadglsentries`).

Since all the sections follow a set format, I'll define a command (`\toxin`) that simply takes a label and displays the complete section. To accommodate the mixture of sections and subsections, we have an associated command (`\toxinsection`) that can be redefined at the start of a chapter where necessary:

```
\newcommand{\toxinsection}{\section}
\newcommand*{\toxin}[1]{%
  \toxinsection{\glsentrytitlecase{#1}{name}}
  \begin{description}
  \item[Toxicity] \toxicity{#1}
  \item[Description] \glsentrydesc{#1}
  \end{description}
}
```

This converts the `name` field to title case in the section heading using `\glsentrytitlecase`, which internally uses `\capitalisewords` provided by the mfirstuc package.

When writing in English, words such as 'and' should only be capitalized when they occur at the start of the title. Since such exceptions are language dependent, they aren't implemented by default. The mfirstuc-english package provides the common English exceptions:

```
\usepackage{mfirstuc-english}
```

This doesn't affect the document so far, but it will later when the 'order by symptoms' list is added.

The main matter is now much shorter:

```
\chapter{Household Poisons}
\renewcommand{\toxinsection}{\subsection}
\section{Chemicals}
\toxin{ammonia}
\section{Food Poisoning}
\toxin{botulinum}
\chapter{Plants}
\renewcommand{\toxinsection}{\section}
\toxin{nutmeg}
\chapter{Street Drugs}
\toxin{lsd}
```

When just considering the main matter, this doesn't seem like a significant improvement to the first example. It is easier to move the sections around, but the title case-changing can't be implemented in the PDF bookmarks. So `\glsentrytitlecase` will expand to the original lowercase value in the bookmark.

It would be useful if the nutmeg reference in the LSD description had a hyperlink to the nutmeg section (created with hyperref). Such hyperlinks are normally achieved with the glossaries package using commands like `\gls`. However, the target anchor is typically in the glossary (implemented by the glossary style), but there isn't a glossary in this document.

The glossaries-extra package provides a solution where you can use `\glsxtrglossentry`:

```
\toxinsection{\glsxtrglossentry{#1}}
```

This command expands to just `\glsentryname` for the PDF bookmark, so there's no difference in this respect. However, within the document text, this command creates the hypertarget and displays the name in the same way as the glossary styles. So I can adjust the case using the glossname attribute:

```
\glssetcategoryattribute{general}{glossname}
                        {title}
```

This only applies to the section title on the page, not in the bookmarks.

The LSD entry can now be modified to include a hyperlink to the nutmeg section:

```
\longnewglossaryentry{lsd}
{name=LSD,toxicity=2}
{Some information about LSD that includes
a reference to \gls{nutmeg}.}
```

There are over two hundred toxins listed in the book. At the moment, all my definitions are stored in my `toxins.tex` file, but I could store them in a `bib` file instead. This would make it easier to share the data across multiple documents. For example, the `bib` file may include brief summaries that can be used as a description in other shorter documents as well as the long description for this catalogue of toxins. For example:

```
@entry{nutmeg,name={nutmeg},
 summary={Short description of nutmeg.},
 longdescription={Long description of nutmeg.}}
```

The document then can choose the appropriate field for the description using field aliases. I'm not going to do this here in order to keep the examples as simple as possible.

My toxins.tex file can easily be converted to toxins.bib using convertgls2bib:

```
convertgls2bib toxins.tex toxins.bib
```

Then I need to replace the code that inputs toxins.tex with:

```
\GlsXtrLoadResources[src=toxins]
```

and add the record package option.

The \latinname command can be provided in the preamble of the bib file to ensure that it's defined:

```
@preamble{
"\providecommand*{\latinname}[1]{\emph{#1}}"}
```

Again, this is something that's useful if the bib file is shared with other documents, but isn't essential for this example as the command is already defined in the document preamble.

Normally, bib2gls will select entries from the bib file if they have records in the aux file (which are created with commands like \gls or \glsadd) or if they depend on selected entries; for example, nutmeg needs to be selected if LSD is selected, since the LSD entry depends on the nutmeg entry.

In this case, though, bib2gls doesn't select any entries because nothing creates a record. (The \gls command in the LSD description will only create a record if the LSD entry is selected and has its description expanded in the document, but there are no LSD records, so LSD won't be selected.)

I could instruct bib2gls to select all entries, but some entries may need to be omitted. For example, the publisher may decide that the print cost for the physical edition is too large, so some entries may need to be dropped.

One approach is to use \glsadd in the definition of \toxin:

```
\newcommand*{\toxin}[1]{%
 \glsadd{#1}% index this entry
 \toxinsection{\glsxtrglossentry{#1}}
 \begin{description}
 \item[Toxicity] \toxicity{#1}
 \item[Description] \glsentrydesc{#1}
 \end{description}
}
```

This ensures that each entry listed in the book will be selected by bib2gls.

The problem of the case-conversion for PDF bookmarks can now be solved as there are some resource options that instruct bib2gls to change the case of field values:

```
name-case-change=title
```

However, this will cause \gls{nutmeg} to start with a capital unless the text field is set to the original value. This can be done with:

```
replicate-fields={name=text}
```

This will copy the value of the name field into the text field. (If the target field is already set, the default behaviour is to leave it unchanged.) Replication is always performed before case-changing, regardless of the resource option ordering. If the source field (name in this case) is not set, the default is to do nothing. But in this case, I want to obtain the value from the fallback if name is missing:

```
replicate-missing-field-action=fallback
```

This means I can now dispense with the glossname attribute.

4 Comma-separated list fields

So far I haven't included the method and symptoms in my entry definitions. I can define two more custom keys in the same way as for toxicity:

```
\glsaddstoragekey{method}{}{\method}
\glsaddstoragekey{symptom}{}{\symptom}
```

I could simply set the values to free-form text:

```
@entry{ammonia,name={ammonia},
 toxicity={4.5},
 description={Some information about ammonia.},
 method={Breathed.},
 symptom={Tachycardia, blindness, lip/mouth
 irritation, burns, flushing, coughing,
 pulmonary edema, abdominal or stomach pain,
 restlessness, collapse, and pain.}
}
```

However, I decided to adopt a different approach. First, I created a file called methods.bib containing:

```
@index{breathed}
@index{injected}
@index{membraneabsorption,
 name={membrane absorption}}
@index{skinabsorption,
 name={skin absorption}}
@index{smoked}
@index{swallowed}
```

The method fields are all going to be comma-separated lists of the method entry labels.

The symptoms are defined in a similar way (in a file called symptoms.bib) but they have an unknown topic field, which will be ignored by bib2gls unless it is aliased or defined in the document (see later). For example:

```
@index{hyperthermia,
 name={fever\MFUwordbreak{\slash}hyperthermia},
 text={fever},
 topic={vital signs}
}
@index{hypothermia,
 name={low body temperature\MFUwordbreak
{\slash}hypothermia},
 text={hypothermia},
```

```
 topic={vital signs}
}
@indexplural{burn,topic={skin}}
@index{collapse,topic={whole body}}
```

Note that while most of these are defined using `@index` there are some defined with `@indexplural`. These entries will default to having the categories set to 'index' and 'indexplural'. This will cause complications later, so all entries will be assigned to the 'general' category with the resource option:

```
category=general
```

The names will be converted to title case so the slash needs to be marked up as a word break using `\MFUwordbreak`, otherwise the word following the slash won't have its case changed. You may prefer to define a string:

```
@string{SLASH="\MFUwordbreak{\slash}"}
```

and use string concatenation:

```
name={fever} # SLASH # {hyperthermia},
```

The ammonia entry can be defined as:

```
@entry{ammonia,name={ammonia},
 toxicity = {4.5},
 description={Some information about ammonia.},
 method = {breathed},
 symptom = {tachycardia,blindness,
mouthirritation,burn,flushing,
coughing,pulmonaryedema,abdominal,
restlessness,collapse,pain}
}
```

The line breaks in the comma-separated lists above can be problematic since these lists will internally be passed to `\@for` in the document, but it is possible to get `bib2gls` to strip the whitespace, if you'd rather not omit them.

There are two options, labelify and labelify-list, that can be used to strip any content that can't occur in a label. The former is intended for fields containing a single label and the latter is for fields containing a comma-separated list of labels. Both are governed by labelify-replace, so the following can be used to strip any whitespace:

```
labelify-list={method,symptom},
labelify-replace={{\string\s+}{}}
```

This means that you can introduce extra space in the `bib` file to make it more readable. Further, since this option also automatically removes empty items, it's also possible to replace ␣and␣ with a comma:

```
labelify-list={method,symptom},
labelify-replace={
 {\string\s+and\string\s+}{,},
 {\string\s+}{}}
```

This means that the list 'A and B' becomes 'A,B'. The list 'A, B, and C' becomes 'A,B,,C'; the empty element is then stripped, leaving 'A,B,C'.

So the `symptom` field can be set as:

```
symptom = {tachycardia, blindness,
  mouth irritation, burn, flushing,
  coughing, pulmonary edema, abdominal,
  restlessness, collapse, and pain}
```

This is not only easier to read but also makes it suitable for use without the `symptoms.bib` file.

The `\glsseelist` command (provided by the base glossaries package) formats a comma-separated list of entry labels. This was designed for the use of cross-referencing with the `see` field [5], but may be used with any list of entry labels. If you want to ensure that the argument is fully expanded, use glossaries-extra's `\glsxtrseelist` instead (which internally uses `\glsseelist`).

The `\toxin` command can be modified to include formatted lists of symptoms, but `\glsseelist` doesn't index so the method and symptom entries won't be selected. In order to ensure that they are selected, `bib2gls` needs to be told that the `method` and `symptom` fields contain lists of dependent entries:

```
dependency-fields={method,symptom}
```

The method and symptom entries don't have any targets (at the moment) so the hyperlinks need to be suppressed. Also the `name` field has had a case-conversion applied. Both problems can be fixed by redefining `\glsseeitem` to just use `\glsentrytext`:

```
\renewcommand*{\glsseeitem}[1]
  {\glsentrytext{#1}}
```

If you want the first item capitalised you can redefine `\glsseefirstitem`:

```
\renewcommand*{\glsseefirstitem}[1]
  {\Glsentrytext{#1}}
```

The separator between the last two items in the list is given by `\glsseelastsep`, which defaults to ␣\&␣. If you want to change this to use 'and' instead:

```
\renewcommand*{\glsseelastsep}{ and }
```

If your preference is for an Oxford comma you will also need:

```
\renewcommand*{\glsseelastoxfordsep}{, and }
```

This may seem a bit redundant since the end result is much the same as the original field value, but the hyperlink will be added in a later example once the corresponding lists have been created.

If you want the method and symptom elements to be alphabetically ordered, then you can instruct `bib2gls` to do this with the sort-label-list option:

```
sort-label-list={
 {method,symptom}:en:glsentryname}
```

This indicates that the method and symptom fields are comma-separated lists and that bib2gls should reorder these lists according to the en sort method (English) where the sort value is obtained by encapsulating the list element with \glsentryname (which bib2gls recognises). This ensures that the list is ordered by the displayed name rather than the label.

5 The index

Since bib2gls sorts by default, a convenient side-effect is that the index can easily be added at the end of the document using the bookindex glossary style (which doesn't show descriptions). As with the other provided glossary styles, this will create a hypertarget, which will cause a conflict, but this can be switched off with the target=false option:

```
\printunsrtglossary[target=false,title=Index]
```

The bookindex style isn't loaded by default, so you'll also need to specify it explicitly:

```
\usepackage[record=nameref,
 stylemods=bookindex,
 style=bookindex]{glossaries-extra}
```

I've set the style as a package option as the other glossaries discussed later will also use this style. If you want letter groups, remember to use the --group (or -g) switch when you invoke bib2gls.

If any entries have the see, seealso or alias fields set, \glsseeitem will need to be restored to its original value for the index. The simplest way to do this is to localise the redefinition. So instead of redefining it in the preamble, it can be redefined within a scoped context within the definition of \toxin. Since environments automatically add scoping, the redefinition can be placed inside the description environment.

Some of the entry descriptions may span multiple pages, in which case you may prefer to have a page range in the index. This can be achieved with explicit location ranges. The position of \glsadd also needs an adjustment. This command switches to horizontal mode (as complications can occur in certain situations otherwise), which means that the page number could be off if the section heading is moved to the start of the next page. If \glsadd is placed after the heading then it will cause an unwanted space before the start of the description environment. The best solution is to place it in the section title and use the optional argument for the bookmark.

```
\newcommand*{\toxin}[1]{%
 \toxinsection[\glsentryname{#1}]{%
\glsxtrglossentry{#1}\glsadd[format=(]{#1}}
 \begin{description}
 \let\glsseeitem\glsentrytext
 \let\glsseefirstitem\Glsentrytext
 \item[Toxicity] \toxicity{#1}
 \item[Method] \glsxtrseelist{\method{#1}}.
 \item[Symptoms] \glsxtrseelist{\symptom{#1}}.
 \item[Description]
\glsentrydesc{#1}\glsadd[format=)]{#1}
 \end{description}
}
```

There is a problem with the method and symptom entries. They haven't been indexed anywhere in the document so they don't have a page list. This could be solved by redefining \glsseeitem to use \glstext instead of \glsentrytext, but this increases the complexity of the document build and could lead to lengthy page lists in the index, especially for common methods (such as swallowing, which applies to most toxins) or symptoms (there's a fairly sizable list for convulsions). Since the final version of this example will have lists of methods and symptoms, there's no need for them to appear in the index.

There are two basic approaches to removing entries from a list: put them in a different glossary or filter them when displaying the list. The first approach can be a bit tricky if all entries are being processed by a single resource command. One way would be to rename toxins.bib to main.bib and use the resource option:

```
type={same as base}
```

This will set the type field to the file basename, without the bib extension. So the entries defined in main.bib will be assigned to the default main glossary, the entries defined in methods.bib will be assigned to the glossary identified by the label methods (which will need to be defined), and similarly entries defined in symptoms.bib will be assigned to a user-provided symptoms glossary.

The second approach keeps all the entries in one glossary but uses the hook that's provided to help skip entries. This is discussed in more detail in a previous article [5], but essentially, in order to avoid problems involved in using iterative code within a tabular-like environment (which some glossary styles use), the entries are first iterated over outside of the glossary and the glossary contents are appended to an internal control sequence. There's a hook that's used in this stage which can skip the current iteration to prevent an entry from being appended.

I've defined a custom command to filter entries with empty locations because it may be useful for other lists (either in this document or placed in a package for the use of other documents):

```
\newcommand{\filteremptylocation}[1]{%
 \glsxtrifhasfield*{location}{#1}
```

```
 {}% has location field
 {\printunsrtglossaryskipentry}%
}
```

The process hook will be `\let` to this command.

Although each entry in the index has a location list, it might be useful to have the entry name as a hyperlink to its section in the main part of the document. The bookindex style provides a command that's used to format the entry name, which takes the entry label as its argument. Again I'm defining a custom command which the style command can locally be `\let` to. This simply encapsulates the name with a hyperlink:

```
\newcommand*{\linkedbookname}[1]{%
 \glshyperlink[\glossentryname{#1}]{#1}}
```

The starred form of `\printunsrtglossary` has a mandatory argument where the code to initialise the hooks can be placed. This is scoped so it won't alter any subsequent lists.

```
\printunsrtglossary*[target=false,title=Index]
{\let\printunsrtglossaryentryprocesshook
\filteremptylocation
 \let\glsxtrbookindexname\linkedbookname}
```

Unfortunately the index now has terms in title case, which doesn't look quite right. I used name-case-change to remove the non-expandable case-changing from the PDF bookmarks, but this has now had an unwanted side-effect. To overcome this problem, I can create a field to store the bookmark title:

```
\glsaddstoragekey{bookmark}{}{\pdfname}
```

Now, instead of copying the `name` to the `text` field, I copy it to this new `bookmark` field:

```
replicate-fields={name=bookmark}
```

and instead of name-case-change I now need to use:

```
field-case-change={bookmark=title}
```

Any formatting commands can be stripped by instructing `bib2gls` to interpret the `bookmark` field:

```
interpret-fields={bookmark}
```

Alternatively, the `\pdfstringdefDisableCommands` command from hyperref can be used to discard problematic tokens.

The `\toxin` command needs to be modified to use this field:

```
\toxinsection[\pdfname{#1}]
 {\glsxtrglossentry{#1}\glsadd[format=(]{#1}}
```

The glossname attribute is needed again, but it has to be switched off before the index. This can be done either by scoping the attribute assignment or by undefining the attribute:

```
\glsunsetcategoryattribute{general}{glossname}
```

6 Synonyms and related terms

The scientific names, alternative names and related substances could be added in a similar way to the symptoms and methods, but it would be useful to have these terms in the index.

There are three cross-referencing fields available [5]: `see`, `seealso` and `alias`. The first two take a comma-separated list of labels. The `alias` field can only have a single label as the value.

I'm going to use the `alias` field for the scientific name, the `see` field for alternative names ('Other') and the `seealso` field for similar substances ('Related'). For example:

```
@index{cbot,
 name={\latinname{Clostridium botulinum}},
 alias={botulinum}}
@index{botulism,see={botulinum}}
@index{botox,see={botulinum}}
@index{lsd-long,
 name={lysergic acid diethylamide},
 alias={lsd}}
@index{lysergide,see={lsd}}
@index{ammoniumhydroxide,
 name={ammonium hydroxide},
 alias={ammonia}}
```

The scientific name for nutmeg is *Myristica fragans*:

```
@index{mfragans,
 name={\latinname{Myristica fragans}},
 alias={nutmeg}}
```

But some other nutmeg species are also listed:

```
@index{margentea,
 name={\latinname{Myristica argentea}},
 alias={nutmeg}}
@index{mmalabarcia,
 name={\latinname{Myristica malabarcia}},
 alias={nutmeg}}
```

This complicates things a little as they need their common name as well:

```
@index{Papuan-nutmeg,
 name={Papuan nutmeg},
 see={margentea}}
@index{Bombay-nutmeg,
 name={Bombay nutmeg},
 see={mmalabarcia}}
```

These new entries aren't referenced anywhere in the document, nor are the selected entries dependent on them, so I need to change the selection criteria to include entries that cross-reference the selected entries:

```
selection={recorded and deps and see}
```

The `\toxin` command needs to be adjusted so that it shows the other names. Entries only have fields that store dependent entries (`see`, etc.), not

the reverse. Whilst it is possible to iterate over all entries to find the synonyms, it's not very efficient.

The resource options save-from-alias, save-from-see and save-from-seealso provide a solution. These define, respectively, the fields `from-alias`, `from-see` and `from-seealso` that contain the required information. The `\toxin` command now needs to check if any of these fields have been defined. For example:

```
\glsxtrifhasfield*{from-see}{#1}{\item[Other]
 \glsxtrseelist\glscurrentfieldvalue}{}
```

The `from-alias` field could be dealt with in the same way, but the common names of the other nutmeg species won't show.

A simple solution is to define a command that checks the `from-see` field that can be used to encapsulate the items in the list:

```
\newcommand{\seeitemandother}[1]{%
 \glsentrytext{#1}%
 \glsxtrifhasfield{from-see}{#1}%
 { (\glsentrytext{\glscurrentfieldvalue})}%
 {}%
}
```

and also an analogous command `\Seeitemandother` that uses `\Glsentrytext` instead, used for the first item in the list. There's a custom command to switch to these commands and display the list:

```
% \toxinitemlist{label}{field}{title}
\newcommand{\toxinitemlist}[3]{%
 \glsxtrifhasfield*{#2}{#1}%
 {\formattoxinitemlist{#3}
                      {\glscurrentfieldvalue}%
 }%
 {}% field not defined
}
```

I've made an inner command to make it easier to adjust the actual formatting:

```
% \formattoxinitemlist{title}{label list}
\newcommand*{\formattoxinitemlist}[2]{%
 \item[#1] {\let\glsseeitem\seeitemandother
  \let\glsseefirstitem\Seeitemandother
  \glsxtrseelist{#2}}.
}
```

The `description` environment within the `\toxin` definition can now be written in a more compact form:

```
\begin{description}
\toxinitemlist{#1}{from-alias}{Scientific Name}
\toxinitemlist{#1}{from-see}{Other}
\toxinitemlist{#1}{from-seealso}{Related}
% other items as before
\end{description}
```

The `\booklinkname` command is now going to cause a problem in the index as these new cross-reference terms don't have a target. There are a number of ways around this. For example, a conditional can be added to use a hyperlink only if the `toxicity` field is set:

```
\newcommand*{\linkedbookname}[1]{%
 \glsxtrifhasfield{toxicity}{#1}%
 {\glshyperlink[\glossentryname{#1}]{#1}}%
 {\glossentryname{#1}}%
}
```

It would, however, be more convenient if the other names could have a hyperlink to their main entry. This could be done by consulting the `alias`, `see` and `seealso` fields in turn, which leads to a complicated set of nested conditionals and also doesn't work for Bombay nutmeg and Papuan nutmeg.

The save-crossref-tail resource option is useful here as it will save the tail label from a cross-reference trail in the `crossref-tail` field. This requires only one extra conditional:

```
\newcommand*{\linkedbookname}[1]{%
 \glsxtrifhasfield{toxicity}{#1}%
 {\glshyperlink[\glossentryname{#1}]{#1}}%
 {%
  \glsxtrifhasfield{crossref-tail}{#1}%
  {\glshyperlink[\glossentryname{#1}]
                {\glscurrentfieldvalue}}%
  {\glossentryname{#1}}%
 }%
}
```

7 Order by toxicity

The glossaries package allows multiple glossaries. A default one is provided with the label `main`. When a new glossary is defined, an internal command is constructed from the name `glolist@`⟨*type*⟩ where ⟨*type*⟩ is the glossary label. Whenever a new entry is defined, its label is appended to the glossary's internal list command's replacement text with a comma separator. It's this list that `\printunsrtglossary` iterates over.

The glossaries-extra package provides a command that copies an entry label to another glossary list. This means that the entry is only defined once and its `type` field is set to its original glossary (this can be considered the entry's primary glossary) but the entry will also appear in the other glossary's list. (This approach can't be used with `makeindex` or `xindy`.)

For example:

```
\newglossaryentry{sample}
 {name=sample,description={}}
\newglossary*{another}{Another}
\glsxtrcopytoglossary{sample}{another}
\begin{document}
\printunsrtglossaries
\end{document}
```

This will display two glossaries, both containing the sample entry. This method allows a duplicate list, which may have a different order, without the overhead of duplicate entry definitions, and it's this method that's employed with `bib2gls`'s secondary resource option.

The syntax for this option is:

`secondary=`⟨*sort*⟩`:`⟨*field*⟩`:`⟨*type*⟩

where ⟨*sort*⟩ indicates the sort method and ⟨*type*⟩ is the glossary label.

The `:`⟨*field*⟩ part is optional and indicates the field to use for sorting. The previous article [7] discussed sorting and, in particular, the system of fallbacks used to determine the value used for comparison if the `sort` field isn't set.

A by-product of the sorting function (regardless of the field used for sorting) is that it assigns the actual sorting value (possibly obtained from fallbacks, with word breaks marked, suffixes appended etc.) to the `sort` field. This means that if you want a secondary glossary, you will need to choose a different field for the sort value unless you want to reuse the same sort values from the primary sort (which would usually be redundant). The secondary sort method will store its actual sort value in the internal field `secondarysort` to avoid conflict, in the event that you need to access both values in your document.

I've used the `--group` switch to add letter groups to my index. You may recall from previous articles that this will store the group label (obtained as a by-product of sorting) in the `group` field. To avoid conflict the secondary sort function will store the group label in the `secondarygroup` internal field, and `bib2gls` will append the required redefinition to the secondary glossary's preamble so that it will automatically switch to the `secondarygroup` field.

So to have a secondary glossary ordered according to the `toxicity` field (from highest to lowest):

```
secondary=integer-reverse:toxicity:bytoxicity
```

Here I've indicated that the secondary glossary has the label `bytoxicity`. If you inspect the `glstex` file, you should find the line:

```
\provideignoredglossary*{bytoxicity}
```

This means that the glossary will be provided if you don't define it; however, it will be an 'ignored' glossary so it will use the default title and won't be picked up by `\printunsrtglossaries`.

If you want to explicitly define this glossary in the document you can add the following (before the resource command):

```
\newglossary*{bytoxicity}{Order by Toxicity}
```

I've used a numeric sort and you may recall from the previous article [7] that this will result in the group label `glsnumbers` (which has the associated language-sensitive title 'Numbers'). This label is actually set indirectly as can be seen from an inspection of the `glstex` file:

```
\bibglssetnumbergrouptitle{{6}{6}{bytoxicity}}
\bibglssetnumbergrouptitle{{4}{4}{bytoxicity}}
\bibglssetnumbergrouptitle{{2}{2}{bytoxicity}}
\bibglssetnumbergrouptitle{{3}{3}{bytoxicity}}
\bibglssetnumbergrouptitle{{0}{0}{bytoxicity}}
```

This command is provided at the start of the file:

```
\providecommand{\bibglssetnumbergrouptitle}[1]{%
 \glsxtrsetgrouptitle
   {\bibglsnumbergroup#1}
   {\bibglsnumbergrouptitle#1}}
```

The group label is obtained from the control sequence `\bibglsnumbergroup` (which must fully expand). This command is also provided:

```
\providecommand{\bibglsnumbergroup}[3]{glsnumbers}
```

It's this definition that causes all entries that have been sorted numerically to be placed in the 'Numbers' group. For this example, I'd like the groups to correspond to the toxicity levels, so I need to define this command before the `glstex` file is input:

```
\newcommand{\bibglsnumbergroup}[3]{#1}
```

The corresponding title is obtained from a command that is also provided in the `glstex` file:

```
\providecommand{\bibglsnumbergrouptitle}[3]{%
 \protect\glsnumbersgroupname}
```

Again I can provide my own definition in the document to override this. For example:

```
\newcommand{\bibglsnumbergrouptitle}[3]{Toxicity
Rating #1}
```

The bookindex headers are formatted according to:

`\glsxtrbookindexformatheader{`⟨*title*⟩`}`

This defaults to a centred format. I've decided to provide a different format, but I don't want to apply it to the index as well, so the redefinition will need to be scoped.

The custom header formatting is quite simplistic for this example:

```
\newcommand{\orderbyheader}[1]{%
 \par{\raggedright\bfseries\large #1\par}}
```

This can be adjusted as required.

As with the index, the list includes entries that don't have the `toxicity` field set. These will end up with a sort value of 0 and can be filtered using a method similar to that employed for the index. However, it's simpler to get `bib2gls` to filter them:

```
secondary-not-match={toxicity={}}
```

The glossary can be displayed in the usual way:

```
\printunsrtglossary[type=bytoxicity,
 target=false]
```

This will include the page list for each entry, which you may prefer to omit. The nonumberlist option can be added to suppress it. On the other hand, it can be useful to have a way to link back to the main definition. As with the index, a hyperlink can be added to the name using the same method as earlier. This is useful for the reader of the PDF version, but for the paperback version it might be helpful to just list the primary page number (rather than the complete list). For this, I've defined a custom location format command:

```
\newcommand{\mainfmt}[1]{\glsnumberformat{#1}}
```

This uses the default formatting, but I can use it to identify the principal page in my custom `\toxin` command:

```
\section[\pdfname{#1}]{\glsxtrglossentry{#1}%
 \glsadd[format=mainfmt]{#1}%
 \glsadd[format=(]{#1}}
```

This primary location format `\mainfmt` is identified with:

```
primary-location-formats=mainfmt
```

I can instruct `bib2gls` to move the primary locations out of the normal location list and into a field called primarylocations:

```
save-primary-locations=remove
```

To ensure that `\printunsrtglossary` uses this field for the location list:

```
\renewcommand{\GlsXtrLocationField}
             {primarylocations}
```

To prevent the primary locations from being merged with the explicit range formation:

```
bib2gls -g --retain-formats mainfmt toxinbook
```

where the document is in the file `toxinbook.tex`.

8 Order by method

The method list is superficially similar. It's not possible to use multiple secondary options in one resource command, but it's possible to have a second resource set that copies entries to another glossary. First define the new glossary:

```
\newglossary*{bymethod}{By Method}
```

Now for the second resource command:

```
\GlsXtrLoadResources[
 src=methods,type=bymethod,
 selection={selected before},
 action=copy
]
```

This selects all the entries in `methods.bib` that were previously selected, sorts them and copies their labels to the `bymethod` glossary. Internally (that is, within `bib2gls`'s Java code) a new object representing each entry is created with the information obtained by reparsing the `bib` file. So any modifications made by the previous resource set won't be present in this resource set (unless the modifications are repeated). This means that the `sort` field will be missing again: the value won't be retained from the previous resource set. However, from LaTeX's point of view, each entry is defined once.

This now provides a target for the methods, so the method list in `\toxin` can have a hyperlink for each entry:

```
\renewcommand*{\glsseeitem}[1]{%
 \glshyperlink[\glsentrytext{#1}]{#1}}
\renewcommand*{\glsseefirstitem}[1]{%
 \glshyperlink[\Glsentrytext{#1}]{#1}}
```

So far this just lists the methods. The sub-list of relevant toxin entries needs to follow each method name. This looks like a hierarchical glossary but it has child entries with multiple parents. The structure is essentially a set of nested glossaries with an outer glossary (the `bymethod` glossary) where each element is followed by an inner glossary.

The glossary process hook can be used to create throwaway glossaries:

```
\let\printunsrtglossaryentryprocesshook
 \provideignoredglossary
```

This creates a glossary with the same label as the entry, but the hook will have to be reverted before the inner glossaries are processed.

There's another hook that's used after processing and just before the glossary is displayed. This can be used to populate the throwaway glossaries and reset the process hook. First a command that does the action:

```
\newcommand{\populatemethods}{%
 \renewcommand
   \printunsrtglossaryentryprocesshook[1]{}%
 \forglsentries[main]{\thislabel}%
 {\glsxtrforcsvfield*{\thislabel}{method}
                     {\populatedo}}%
}
```

with a list handler:

```
\newcommand{\populatedo}[1]{%
 \glsxtrcopytoglossary{\thislabel}{#1}%
}
```

Then the hook needs to be assigned to this command within a scoped context:

```
\let\printunsrtglossarypredoglossary
    \populatemethods
```

The glossary entry handler needs to not only display the current entry (as it does by default with `\glsxtrunsrtdo`) but also follow it with an inner glossary. First the custom nested handler that takes the current entry label as the argument:

```
\newcommand{\nestedhandler}[1]{%
 \glsxtrunsrtdo{#1}%
 \ifglossaryexists*{#1}%
 {%
  \printunsrtinnerglossary
    [type={#1},groups=false,target=false]
  {%
   \let\glsxtrbookindexname\linkedbookname
   \renewcommand{\GlsXtrLocationField}
                {primarylocations}%
  }{}%
 }%
 {}%
}
```

This tests if there's a glossary with a label matching the entry's label and, if it exists, that glossary will be displayed (but without the title). As with the ordered by toxicity glossary, only the primary location is displayed and the name is hyperlinked to easily jump back to the main definition.

All glossaries are using the bookindex style, and since the method and toxin entries are all top-level entries (they don't have a parent), they will all end up with their names formatted in the same way (as a top-level entry). The method entries need to have their names formatted in the same way as the group titles to be consistent with the order by toxicity glossary.

```
\newcommand{\prenamesep}{}
\newcommand{\orderbyname}[1]{%
 \prenamesep
 \glsxtrbookindexbookmark{\pdfname{#1}}
   {\glsxtrbookindexbookmarkprefix#1}%
 \orderbyheader{\glossentryname{#1}}%
 \def\prenamesep{\par}%
 \par\smallskip
}
```

(Again, the style is simplistic to reduce the complexity of the example.) It's then possible to switch to this in a scoped context:

```
\let\glsxtrbookindexname\orderbyname
```

9 Order by symptoms

The symptoms list is more complicated as it's divided into different categories: 'vital signs', 'head, eyes, ears, nose, throat', 'skin', 'heart' and so on. These correspond to the custom `topic` field that has so far been ignored by `bib2gls`. Note that these topics aren't listed in alphabetical order. Within each topic is a sub-list of symptoms, which is ordered alphabetically. I'm first going to start off with the topics alphabetically ordered and then make an adjustment to achieve the desired result.

The topics and their sub-lists are essentially hierarchical, so the `topic` field can be aliased to `parent`. The labelify option can strip the spaces using the same labelify-replace setting used earlier.

```
field-aliases={topic=parent},
labelify={parent}
```

The parent entries (representing the topics) need to be defined, so I've created a new file called `topics.bib` that contains:

```
@index{vitalsigns,name={vital signs}}
@index{head,
 name={head, eyes, ears, nose, throat}}
@index{skin}
@index{heart}
@index{airway,
 name={airway and lungs}}
@index{gastrointestinal,
 name={gastrointestinal system}}
@index{fluids,
 name={fluids and electrolytes}}
@index{neurological,
 name={neurological system}}
@index{psychiatric}
@index{wholebody,
 name={whole body and miscellaneous symptoms}}
```

This file needs to be added to the src list:

```
src={toxins,methods,symptoms,topics}
```

Since child entries depend on their parent, the parent entries will automatically be selected when the child entry is selected. The topics and symptoms can be copied to a new glossary in the same way as the methods:

```
\newglossary*{bysymptoms}{By Symptoms}
\GlsXtrLoadResources[
 src={symptoms,topics},
 type=bysymptoms,
 selection={selected before},
 action=copy,
 field-aliases={topic=parent},
 labelify={parent},
 labelify-replace={{\string\s+}{}}
]
```

The process hook is similar to the hook used for the method list but the throwaway glossaries are only created for child entries:

```
\newcommand{\symptomsprocesshook}[1]{%
 \ifglshasparent{#1}%
 {\provideignoredglossary{#1}}%
 {}%
}
```

The code to populate the symptom glossaries is similar to that used for the methods:

```
\newcommand{\populatesymptoms}{%
 \renewcommand
\printunsrtglossaryentryprocesshook[1]{}%
 \forglsentries[main]{\thislabel}%
 {%
```

```
  \glsxtrforcsvfield*{\thislabel}{symptom}
{\populatedo}%
 }%
}
```

The entry handler is the same one as used before (`\nestedhandler`).

The formatting of the sub-items (the symptom entries) needs adjusting. The bookindex style formats the sub-item names according to:

`\glsxtrbookindexsubname{`⟨*label*⟩`}`

The default definition uses `\glsxtrbookindexname` so this will need to be changed. First I need a custom sub-header to match the headers used in the toxicity and method lists:

```
\newcommand{\orderbysubheader}[1]{%
 \par{\raggedright\bfseries #1\par}}
```

Again this is simplistic, to be modified as required. The custom command for child entry names is:

```
\newcommand{\orderbychild}[1]{%
 \pdfbookmark[2]{\pdfname{#1}}
   {\glsxtrbookindexbookmarkprefix#1}%
 \orderbysubheader{\glossentryname{#1}}%
 \par\smallskip
}
```

I added the mfirstuc-english package earlier, but up until now it hasn't been needed. This package can be implemented by `bib2gls` but isn't by default. In order to ensure that the title-case word exceptions provided by that package are used by `bib2gls`, it's necessary to use the `--packages` (or `-p`) switch:

```
bib2gls -g --retain-formats mainfmt \
  --packages mfirstuc-english toxinbook
```

Finally, I want to have the topics listed in the order that they are defined in the `topics.bib` file. This is quite awkward as it's not possible to apply a different sort method to each hierarchical level. However, it is possible to encapsulate the sort value (after it has been obtained from fallbacks and any other processing, such as word breaks and suffixes). The encapsulation command must take two arguments: the first is the sort value that has been determined so far, and the second is the entry's label.

It's possible to save the entry definition index using save-definition-index. With this setting, the definition index can be accessed with the command `\bibglsdefinitionindex`. The aim here is to define a command that finds out if an entry has a parent. If it has, then the ordinary sort value is used. If it hasn't, then the definition index is used instead. Since the sort method is alphabetical, the definition index will need to be zero-padded to ensure that it's correctly ordered. Here I've padded up to six digits, which should be ample:

```
format-integer-fields={definitionindex=\%06d}
```

My custom command is provided in the preamble of `topics.bib`:

```
@preamble{"\providecommand{\topicsort}[2]{%
\ifglshasparent{#2}{#1}
                  {\bibglsdefinitionindex{#2}}}"}
```

This command now needs to be specified as the sort encapsulator:

```
encapsulate-sort=topicsort
```

There's no need for this preamble to be written to the `glstex` file since it's not required in the document:

```
write-preamble=false
```

10 Duplicate entry

Stevens and Bannon's book lists 'botulism' in the 'Household Poisons' chapter and 'botulism toxin' in the 'Biological, Chemical and Radiological Weapons' chapter. The two entries have different descriptions, so they need to be defined as two separate entries, but it would be strange to have two 'botulinum' entries listed in the index.

One solution is to change the botulinum entry to a dual entry:

```
@dualentry{botulinum,name={botulinum},
 toxicity = {6},
 description={Some information about foodborne
botulism.},
 dualdescription={Some information about the
botulinum toxin used as a bioweapon.},
 method={injected and swallowed},
 symptom={blurred vision, nausea and paralysis}
}
```

This defines two linked entries. The primary entry has the label `botulinum` and the dual entry has the label `dual.botulinum`. The default is to swap the name and descriptions around in the dual and copy all the other fields. In this case I want to keep the names the same but switch descriptions, which can be done with the resource option:

```
dual-entry-map={{dualdescription},{description}}
```

Since I don't need a separate list of dual entries, it's more efficient to combine the dual into the primary list:

```
dual-sort={combine}
```

I also want to move all the dual locations over to the primary entry's location list:

```
combine-dual-locations={primary}
```

This means that the original 'botulinum' entry will appear in the index with its own locations merged with the dual locations. Since the dual entry's location list ends up empty, the filter will exclude it so it won't appear in the index. The filter is also needed in the other lists. The toxicity list:

```
\printunsrtglossary*[type=bytoxicity,
                     target=false]
{%
 \let\glsxtrbookindexformatheader\orderbyheader
 \let\glsxtrbookindexname\linkedbookname
 \let\printunsrtglossaryentryprocesshook
     \filteremptylocation
 \renewcommand{\GlsXtrLocationField}
              {primarylocations}%
}
```

For the symptoms and method lists, the filter needs to be in the inner glossary:

```
\newcommand{\nestedhandler}[1]{%
 \glsxtrunsrtdo{#1}%
 \ifglossaryexists*{#1}%
 {%
  \printunsrtinnerglossary[type={#1},
                     groups=false,target=false]
  {%
   \let\glsxtrbookindexname\linkedbookname
   \let\printunsrtglossaryentryprocesshook
       \filteremptylocation
   \renewcommand{\GlsXtrLocationField}
                {primarylocations}%
  }{}%
 }%
 {}%
}
```

Notice that the cross-reference fields only reference the primary entry, not the dual. This means that the scientific name and other names will be missing for the dual entry. However, it's possible to adjust the definition of `\toxinitemlist` so that it fetches the information from the primary entry. In order to do this it's first necessary to instruct `bib2gls` to save the label of the opposite entry for dual entries. This can be done with the dual-field resource option, which will save the label of the opposite entry in the `dual` field. This means that the `dual.botulinum` entry will have the `dual` field set to `botulinum` and the `botulinum` entry will have the `dual` field set to `dual.botulinum`.

```
\newcommand{\toxinitemlist}[3]{%
 \glsxtrifhasfield*{#2}{#1}%
 {%
  \formattoxinitemlist{#3}
                      {\glscurrentfieldvalue}%
 }%
 {%
  \glsxtrifhasfield*{dual}{#1}%
  {% is a dual entry
   \glsxtrifhasfield*{#2}
                     {\glscurrentfieldvalue}%
   {%
     \formattoxinitemlist{#3}
                     {\glscurrentfieldvalue}%
   }%
   {}%
  }%
  {}% not a dual entry
 }%
}
```

The complete document and `bib` files can be downloaded [4]. I've used the uelem package and hyperref's hidelinks option, so the hyperlinks show up underlined, to make them visible in the printed figures here. The key command for that:

```
\renewcommand{\glsxtrhyperlink}[2]{%
  \hyperlink{#1}{\uline{#2}}}
```

The standalone entries are shown in figure 1 (household poisons — ammonia and the primary botulinum entry), figure 2 (plants — nutmeg), figure 3 (street drugs — LSD) and figure 4 (biochemical warfare — dual botulinum entry).

Figure 5 shows the toxicity list, figure 6 shows the methods list, figure 7 shows the first page of the symptoms list, and figure 8 shows the index. Finally, figure 9 shows the PDF bookmarks (in Okular).

References

[1] F. Mittelbach. The multicol package, 2021. ctan.org/pkg/multicol.

[2] S. Rahtz, H. Oberdiek, The LaTeX3 Project. The hyperref package, 2021. ctan.org/pkg/hyperref.

[3] S. Stevens, A. Bannon. *Book of Poisons: A guide for writers*. Howdunit. Writer's Digest Books, 1st ed., 2007.

[4] N. Talbot. Sample `bib` files. dickimaw-books.com/latex/tugboat-bib2gls.

[5] N. Talbot. `bib2gls`: selection, cross-references and locations. *TUGboat* 41(3), 2020. tug.org/TUGboat/tb41-3/tb129talbot-bib2gls-more.pdf.

[6] N. Talbot. `bib2gls`: Command line application to convert `.bib` files to `glossaries-extra.sty` resource files, 2021. ctan.org/pkg/bib2gls.

[7] N. Talbot. `bib2gls`: sorting. *TUGboat* 42(2), 2021. tug.org/TUGboat/tb42-2/tb129talbot-sorting.pdf.

[8] N. Talbot. The glossaries-extra package, 2021. ctan.org/pkg/glossaries-extra.

[9] N. Talbot. The glossaries package, 2021. ctan.org/pkg/glossaries.

[10] N. Talbot. The mfirstuc package, 2021. ctan.org/pkg/mfirstuc.

◇ Nicola L. C. Talbot
School of Computing Sciences
University of East Anglia
Norwich Research Park
Norwich NR4 7TJ
United Kingdom
https://www.dickimaw-books.com

Chapter 1

Household Poisons

Chemicals

Ammonia

Scientific Name Ammonium hydroxide.

Toxicity 4.5

Method Breathed.

Symptoms Abdominal or stomach pain, blindness, burns, collapse, coughing, flushing, mouth irritation, pain, pulmonary edema, restlessness, and tachycardia.

Description Some information about ammonia.

Food Poisoning

Botulinum

Scientific Name *Clostridium botulinum*.

Other Botox and botulism.

Toxicity 6

Method Injected and swallowed.

Symptoms Blurred or double vision, nausea, and paralysis.

Description Some information about foodborne botulism.

Figure 1: Standalone entries: Household Poisons

Chapter 2

Plants

Nutmeg

Scientific Name *Myristica argentea* (Papuan nutmeg), *Myristica fragans*, and *Myristica malabarcia* (Bombay nutmeg).

Toxicity 3

Method Injected and swallowed.

Symptoms Anxiety, blurred or double vision, convulsions, dehydration, dry mouth, euphoria, fever, flushing, hallucinations, irregular heartbeat, nausea, pain, psychosis, and tachycardia.

Description Some information about nutmeg.

Figure 2: Standalone entries: Plants

Chapter 3

Street Drugs

LSD

Scientific Name Lysergic acid diethylamide.

Other Lysergide.

Toxicity 2

Method Injected and swallowed.

Symptoms Coma, confusion, convulsions, excitement, hallucinations, psychosis, and spasms.

Description Some information about LSD that includes a reference to nutmeg.

Figure 3: Standalone entries: Street Drugs

Chapter 4

Biochemical Warfare

Botulinum

Scientific Name *Clostridium botulinum*.

Other Botox and botulism.

Toxicity 6

Method Injected and swallowed.

Symptoms Blurred or double vision, nausea, and paralysis.

Description Some information about the botulinum toxin used as a bioweapon.

Figure 4: Standalone entries: Biochemical

Order by Toxicity

Toxicity Rating 6

Botulinum, 1, 7

Toxicity Rating 4

Ammonia, 1

Toxicity Rating 3

Nutmeg, 3

Toxicity Rating 2

LSD, 5

Figure 5: Order by toxicity

By Method

Breathed

Ammonia, 1

Injected

Botulinum, 1, 7
LSD, 5
Nutmeg, 3

Swallowed

Botulinum, 1, 7
LSD, 5
Nutmeg, 3

Figure 6: Order by method

By Symptoms

Vital Signs

Fever/Hyperthermia

Nutmeg, 3

Tachycardia/Rapid Heartbeat or Pulse

Ammonia, 1
Nutmeg, 3

Head, Eyes, Ears, Nose, Throat

Blindness

Ammonia, 1

Blurred or Double Vision

Botulinum, 1, 7
Nutmeg, 3

Dry Mouth

Nutmeg, 3

Lip/Mouth Irritation

Ammonia, 1

Skin

Burns

Ammonia, 1

Flushing/Turning Red

Ammonia, 1
Nutmeg, 3

Heart

Irregular Heartbeat

Nutmeg, 3

Airway and Lungs

Coughing

Ammonia, 1

Pulmonary Edema

Ammonia, 1

Gastrointestinal System

Abdominal or Stomach Pain

Ammonia, 1

Figure 7: Order by symptom

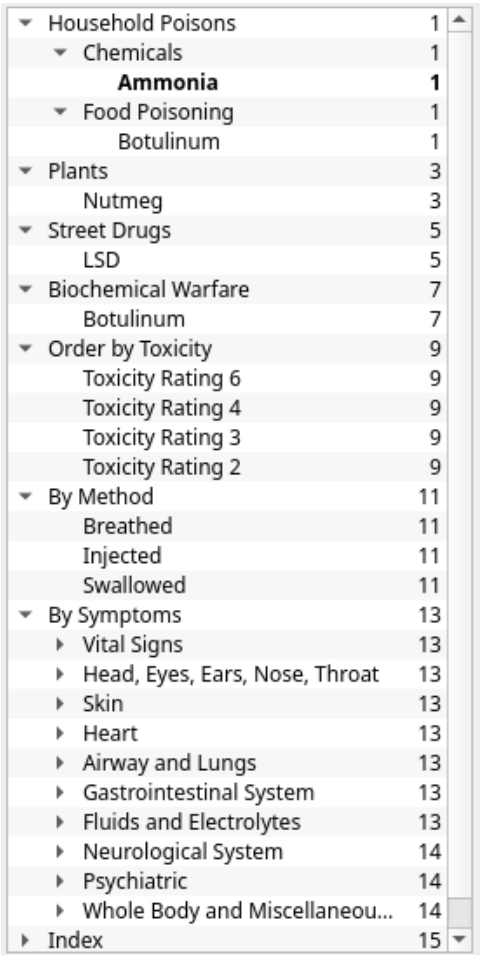

Figure 9: PDF bookmarks

Index

Figure 8: Index